RESILIENCE-ENHANCED CONTROL RECONFIGURATION FOR AUTONOMOUS SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Sehwan Oh

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Aerospace Engineering

Georgia Institute of Technology

August 2019

RESILIENCE-ENHANCED CONTROL RECONFIGURATION FOR AUTONOMOUS SYSTEMS

Approved by:

Prof. Dimitri N. Mavris, Advisor
School of Aerospace Engineering
Georgia Institute of Technology

Prof. Daniel P. Schrage
School of Aerospace Engineering
Georgia Institute of Technology

Prof. George J. Vachtsevanos
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Scott Duncan
School of Aerospace Engineering
Georgia Institute of Technology

Dr. Jean C. Domercant
Electronic Systems Laboratory
Georgia Tech Research Institute

Date Approved: June 21, 2019

According to Darwin's Origin of Species, it is not the most intellectual of the species that survives; it is not the strongest that survives; but the species that survives is the one that is able best to adapt and adjust to the changing environment in which it finds itself.

Megginson, 1963

To my mother, my father, and my wife,

Thank you.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to my adviser, Dr. Dimitri Mavris, for his support, advice, and guidance. His dedication and insight have always been my role model, and he will always be remembered. I am thanking Dr. George Vachtsevanos for his academic guidance, critical inputs, and an enormous help. Appreciation is also expressed to the committee members, Dr. Daniel Schrage, Dr. Scott Duncan, and Dr. Jean Domercant. Their comments and suggestions contributed to improving the quality of this work.

I also want to express the deepest appreciation to my parents, my wife, and my kids. Their endless support and love made me come to this far. It would never be possible without them.

Last but not least, I am grateful to all my friends in ASDL, school, and societies. Their kindness and generosity will never be forgotten.

Thank you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis presents a resilience-enhanced reconfigurable control framework for unmanned autonomous systems in the presence of a critical fault mode during the operation. Such a critical fault mode significantly impairs system performances and long-term mission capabilities. If it is not adequately treated, system and mission failures are inevitable. This research assumes that 1) mission and mission profile are given, 2) a fault grows monotonically, 3) fault indicators are definable, 4) multiple fault modes do not coincide, 5) measurements regarding fault indicators are online, and 6) system knowledge and historical data are accurate and obtainable.

This research has been motivated by the safety concerns that prevent Unmanned Autonomous Systems (UAS) from active introductions to practical applications. Various sources predicting unmanned system market values suggest significant growth in not only the number of UAS applications, but also financial benefits on industries. However, any delay in the realization of UAS applications will cause a loss of the opportunity cost.

According to the Unmanned Aerial Vehicle (UAV) incidence reports, mechanical failures come out to be one of the top reasons for the incidents except for human errors. Technically, it is impossible to avoid any fault or failure in any systems. However, it can be possible to save the faulty system if the faults are treated properly.

In this regard, this research has reviewed the state-of-the-art techniques regarding system safety improvement in the presence of a critical fault mode and identified research opportunities by gap analyses. Promising concepts that have been extensively reviewed are resilience engineering and Active Fault Tolerant Control (AFTCS) systems. Resilience engineering has been more focus on system design and resilience assessment methods. AFTCS mainly contributes to the fast and stable operating point recovery without the consideration of long-term system performances. Prognostics-enhanced reconfigurable control frameworks have proposed the online prognosis within the control scheme but do not ad-

dress comprehensive mission capability trade-offs.

The proposed resilience-enhanced reconfigurable control framework is composed of three fundamental modules: 1) immediate performance recovery by Model Predictive Control (MPC) and Differential Dynamic Programming (DDP) approaches, 2) long-term mission capability trade-offs by an optimization routine, and 3) situational awareness by a particle filtering-based fault diagnosis and case-based reasoning. Critical development of this thesis is an introduction of an adaptation parameter in an MPC formulation (Module 1) and optimization process to find an optimal value for the adaptation parameter (Module 2). Module 3 enables long-term mission capability reasoning when a new fault growth pattern is observed. A set of research questions and hypotheses helps form the technical approach and a grand experimentation plan.

In order to test the efficacy of the proposed framework, under-actuated hovercraft as a testbed and an insulation degradation of an electrical thrust motor as a critical fault mode are introduced. The experiments explore the effect of the adaptation parameter on long-term mission capabilities and identify the necessity of the proper trade-offs. Further experiments investigate the efficacy of each module and the integrated framework. The experiment results show that the adaptation parameter adjusts a control strategy, so that mission capabilities are optimized while vulnerable long-term mission capabilities are recovered. The integrated framework presents the improvement to the probability of mission success in the presence of a critical fault mode.

CHAPTER 1
INTRODUCTION

Unmanned systems keep replacing manned systems as a paradigm shift. According to the Unmanned Autonomous Systems (UAS) market forecast reports, the UAS market value is expected to grow two to three times higher in ten years [1, 2, 3, 4, 5]. Figure 1.1 and 1.2 are UAV market forecast reports for military and commercial applications, respectively. Considering the economic impacts of UAS application in job markets and component manufacturing industries, the UAS market value may very well exceed, which is predicted in the reports. Unmanned Aerial Vehicles (UAVs) have already been deployed and are actively performing in military operations. Some of the notable UAVs in the military industry include Hunter and Shadow of the US Army, Pioneer and Fire Scout of the US Navy, and Predator and Global Hawk of the US Air Force. Their various missions range from reconnaissance to armed attack. Further development of UAVs is expected to enhance the mission profiles to even more agile tasks such as combats and search & rescue in complex environments. Besides, Unmanned Surface and Underwater Vehicles (USVs and UUVs) are also developed and introduced in military applications. UASs are also up-and-coming systems for industrial purposes such as commercial surveillance, agriculture, scientific research, journalism, and cargo transportation. There is no doubt that UAS will play critical roles in various missions.

However, regulations have limited the effective utilization of UAS due to safety concerns. Federal Aviation Administration (FAA) announced new rules for drones in 2016, which restrict operations only for small UAVs under 55 pounds to line-of-sight visibility and daylight-only flights [6].

These restrictive regulations significantly delay the potential usefulness of civilian and commercial UAVs. Larry Downes, who is a co-author of Big bang disruption, stated that

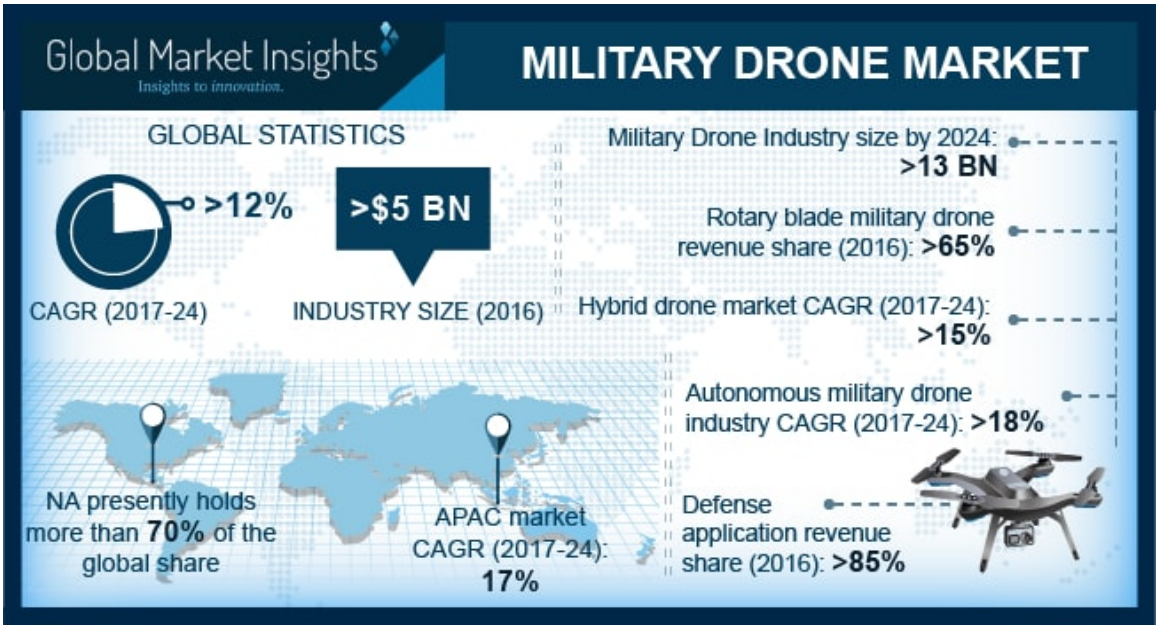

Figure 1.1: Military drone market forecast [4]

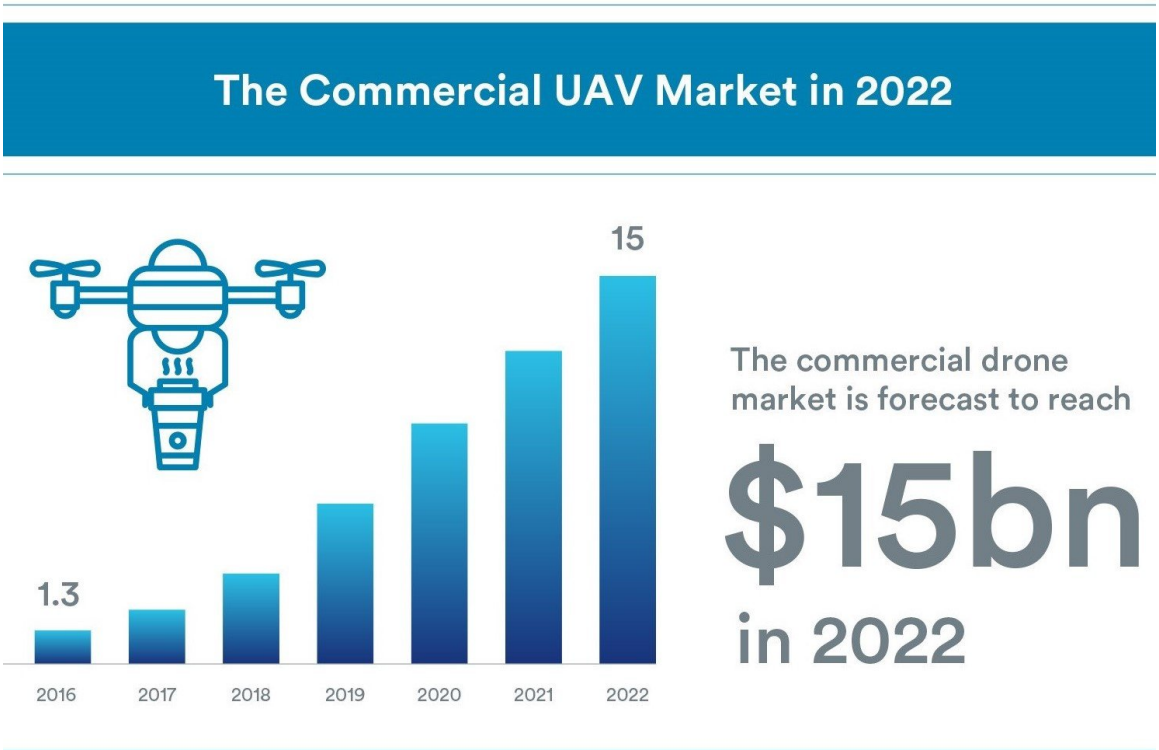

Figure 1.2: Commercial drone market forecast [5]

[7]:

> "When it comes to flying electronic devices weighing up to 55 pounds, safety is an essential requirement. But are the delays justified? The short answer is no. The proposed rules are vague and incomplete, where they could easily be straightforward and even obvious. Special rules for micro-drones should have come first, not just hinted at last. And an unnecessarily restrictive requirement that all drone operation requires continual 'line of sight' visibility and daylight-only operation means that some of the most high-potential applications, including local delivery services and nighttime agricultural monitoring, are still banned."

Such restrictions in UAS application are mainly due to safety concerns. Extreme disturbances in the middle of operations can cause system/mission failures, which ultimately can result in the loss of the physical system, intelligence leakage, collateral damages to ground structures, or even human lives. According to an investigation report in 2016 by the Washington Post, 236 military drone crashes were classified as Class A mishaps since 2001, which are accidents that caused at least $2 million loss [8]. One article from the Washington Post pointed out concerns regarding exposure of secret US military operation strategies due to drone crashes in hostile territories [9]. Another article stated that a Reaper crashed near a densely populated area in San Diego during a surveillance mission in 2014 [10].

Due to confidentiality issues, the details of the causes of military UAV accidents have not been fully disclosed. Several investigations, however, reported that the primary causes were pilot errors and mechanical failures [11, 12]. As human intervention decreases with increasing unmanned systems, mechanical failures will be one of the top culprits of UAV crashes.

Mechanical faults/failures - particularly in vehicle-type systems - affect overall system performances such as system stability, controllability, maneuverability, duration, and

ranges. For example, suppose that one UAV is given a mission: fly from point A to point B with a fixed cruise speed setting. Suppose that the UAV takes off from point A, and a fault occurs in one actuator on the way to point B, resulting in thrust power loss and control asymmetry. If the thrust power loss is not significant, then the UAV could still perform the given mission by a closed-loop controller. However, if the control asymmetry becomes severe, then the corresponding performance degradation will not be ignorable, possibly causing critical events such as mission failure and system loss. Unfortunately, no one can guarantee that the fault/failure events never happen during the operation.

The consequences of critical faults and failures are not acceptable. There will be a system loss for sure. Due to the system loss, a mission could fail. The loss and failure of a system and mission will highly likely cost a considerable amount of money. Collateral damages would add costs. Also, such UAV's are a collection of knowledge in multi-disciplines; thus, the loss of such systems could lead to the leakage of proprietary intellectual properties.

To alleviate this issue, improving robustness could increase system reliability and decrease system sensitivity to component fault/failure events. In definition, robust design means a design concept of optimizing system performances under various disturbance scenarios [13]. Simply speaking, it could be done by 1) improving system reliability, and 2) introducing hardware redundancies, or both.

However, making a 100-percent reliable system is undoubtedly impossible. Instead, future engineering systems would have more chances of experiencing more undesirable situations due to the growing need for advanced mission capabilities [14]. Shelton described the increase in capability requirements and correspondingly more capable systems in the US military [15]. In order to cope with the increase in capability requirements, a system needs more subsystems, components, and their interconnections serving the growing requirements. Sometimes, even multiple systems for a single mission may be necessary. As such, the increase in the mission capability requirements inherently drives the system com-

plexity higher [16]. Moreover, increasing system complexity results in more operational performance uncertainty [13].

Moreover, introducing redundancy is not a desirable option. Recent developments of UAS tend toward lighter and smaller systems. Significant weight and volume requirements constrain the design of such small systems, and there is little affordable space for the redundant subsystems.

Such difficulties have led to active safety management techniques for critical faults and failures. A Fault Tolerant Control System (FTCS) is a control system that can tolerate undesirable fault effects and improve system reliability and survivability.

Resilience engineering is another research thread that has been recently highlighted as a promising concept for safety improvement [13]. Resilience is a bio-inspired idea initially introduced in ecological systems by Holling [17], where the essential elements are intelligent decisions and actions by observing environments and anticipating what will happen. In engineering, the concept of resilience is still evolving, but there is no doubt that it has excellent potentials if it is appropriately applied.

Under the motivations, the objectives of this research are:

Ultimate Research Goal

- To develop a design methodology of a control framework that can intelligently alleviate the impact of critical fault modes during operations and manage the degraded system to complete its mission before failure
- To embrace the concept of resilience in the control framework. A proactive and comprehensive mission capability recovery under threats is a critical concept to embrace.

1.1 Research contributions

A brief list of research contributions was summarized below:

1. An introduction to the concept of resilience in AFTCS
2. Consideration of the impact of adaptation on mission capabilities
3. An introduction to an adaptation parameter
4. A resilience-enhanced reconfigurable control framework
5. MPC-DDP as AFTCS
6. Simulation-based mission capability modeling
7. CBR for unknown fault growth models
8. Under-actuated hovercraft and thrust motor fault example.
9. Demonstration of resilience improvement
10. Design methodology for the proposed framework
11. Demonstration of test cases in Matlab

1.2 Dissertation structure

This dissertation is organized into six chapters. Chapter 1 has introduced the research motivation. From the motivation, research problems, objectives, and research contributions were identified.

Chapter 2 reviews literature regarding fault & failure, fault diagnosis & failure prognosis, resilience engineering, and FTCS. Three research gaps have been identified from the state-of-the-art techniques of resilience engineering and FTCS. The gaps have posed a focused research problem and research objectives.

Chapter 3 proposes technical approaches addressing the research problem. Three submodules and the integrated framework are described. As deriving the alternatives of technical approaches, research questions have been defined for each module and the framework. Along with the research questions, testable hypotheses have raised to prove the effectiveness and efficacy of the proposed methods.

Based on the hypotheses defined in Chapter 3, Chapter 4 demonstrates grand experimentation plans and test results accordingly. Under-actuated hovercraft and insulation degradation in an electric thrust motor were introduced as a testbed. Each module was assessed one-by-one by the tests, and corresponding hypotheses were rejected or not rejected based on results and observations. It finalizes the technical methods for the proposed framework.

Chapter 5 described the design methodology of the proposed framework. Assumptions were categorized to identify the requirements and limitations of the proposed framework.

Chapter 6 concludes by summarizing research objectives and technical findings throughout the research. The observations suggest possible future works that can advance the proposed framework into a deeper embrace of resilience and practical applicability.

CHAPTER 2
BACKGROUNDS

2.1 Fault and failure

Fault and failure for engineering systems are distinguishable terms in diagnostics/prognostics as shown below:

- **Definition: Fault**
  - An unpermitted deviation of at least one characteristic property or parameter of the systems from the normal conditions [18, 19]
  - An unpredicted or unexpected change in systems, such as component malfunction and variations in operating condition, tending to degrade overall system performance [20, 21]
- **Definition: Failure**
  - A permanent interruption of the system ability to perform a required function under specified operating conditions [19]

According to the definitions above, faults are in the physical domain, whereas failures lie in the functional domain. For instance, at a system design phase, designers usually conduct functional decomposition of a system to meet the design requirements. The requirements hierarchically flow down as top functions are decomposed into sub-functions. At a certain level of functional decomposition, sub-functions require to choose physical subsystems or control algorithms performing the corresponding sub-functions. Then, the subsystems or control algorithms derive sub-functions again until physical subsystems are decomposed into very low-level components. From this perspective, faults are the root

causes of functional degradation, possibly resulting in failures. Failures, on the other hand, imply a complete loss or unacceptable degradation of specific functionality. Thus, failures are the consequences of faults. The effects of faults are supposed to be captured in a measurement domain as "symptoms." In other words, when a physical subsystem is faulty, the corresponding functionality suffers, and the effect is shown through detailed measurements.

In this context, the existence of a fault does not necessarily mean a total loss of certain system functionality. Instead, a minor or an early stage of fault development may be acceptable during system operation. As an analogy, people can live with a minor or an early development stage of diseases. For example, in case of an outbreak of a tumor, people may not even know they have it because organs function just fine with little effect from the tumor. Maybe treatments are not necessary if the tumor does not turn out to be malignant. However, if the tumor transforms into a form of cancer and grows, organs affected by cancer will not be the same as before. Their functionality will be severely degraded, which can cause gradually growing pain as a type of symptoms so that the patient feels that something is wrong with his/her health. If the disease is not treated the right way at the right time, the organs will eventually collapse, and the effect may even cause death.

This disease analogy can be correctly translated into the concept of the fault and failure from the engineering point of view as shown in Figure 2.1. For instance, the initiation of a tiny crack on the surface of bearing support is a fault and does not cause an immediate loss of bearing functionality, but only negligible vibrations as a symptom. If the crack is worn out by bearing balls during the operation, or the vibrations are within the acceptable level, there will be no need of fixing or replacing the support. On the contrary, if the crack keeps propagating as the bearing operates, vibrations will get severe, and finally, the bearing will be no longer usable due to the bearing support failure.

Fault and failure can be categorized by their characteristics. Failure Modes and Effects Criticality Analysis (FMECA) published in MIL-STD-1629A [22] and BS 5760-5 [23] provide a procedure about how to rank possible fault and failure modes based on their

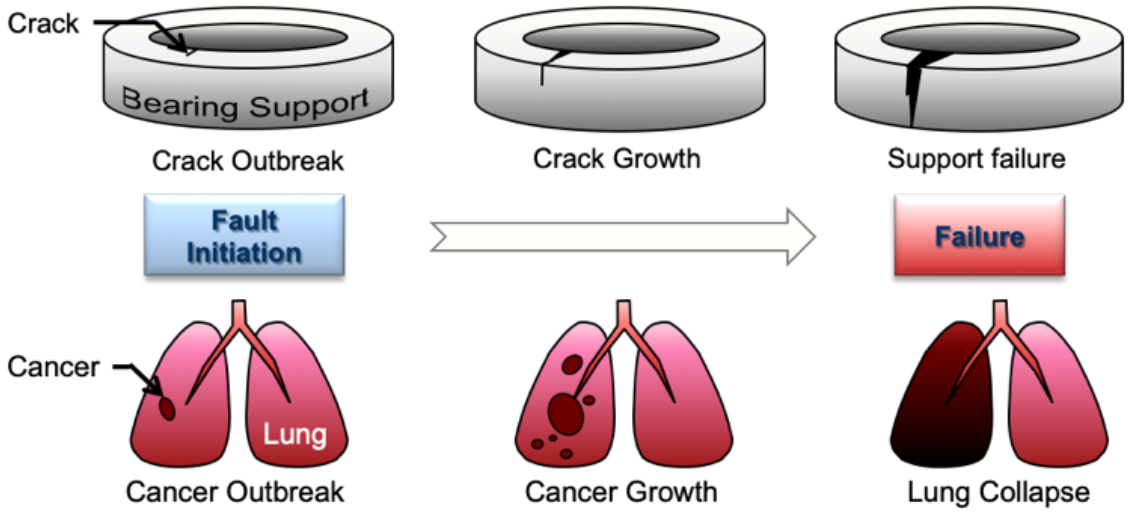

Figure 2.1: Analogy of fault growth to disease

effects and consequences. There are three categories to support the ranking procedure as followings [22]:

1. **Severity classification**: Severity is a level of consequences of failure effects on a total system. MIL-STD-1629A suggests four categories: minor, marginal, critical, and catastrophic.

2. **Frequency of occurrence**: Frequency of occurrence is characterized by a level of the failure rate of a subsystem or a component. Mean-time-to-failure or probability of occurrence are means of quantitative or qualitative measures, respectively.

3. **Detectability**: Detectability is characterized by the probability of detection of defects based on the current design of a system.

Category 1 and 2, severity and frequency of failure modes, can define criticality as shown in Figure 2.2. Criticality increases as failures become severe and frequent. BS 5760-5 suggests a means of a quantitative measure of the failure criticality. Category 3, on the other hand, shows whether a current configuration of a system is capable of monitoring specified failures. If a particular defect causing catastrophic results is not detectable,

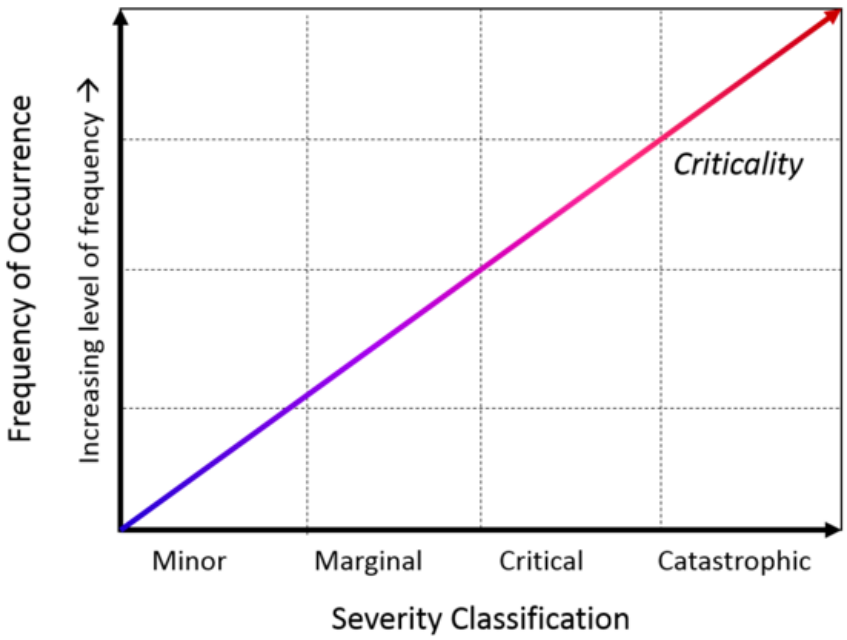

Figure 2.2: Example of criticality matrix [22]

radical design changes of the system may be necessary. Finally, based on the ranking and detectability of failures, detectable faults causing critical consequences are chosen.

Moreover, faults behave in different ways. It implies that a fault diagnosis and failure prognosis framework aims at monitoring specific types of faults and monitors them based on their characteristics. Faults can also be categorized by their characteristics as described in the reference [24]:

1. **Time behavior point of view**

   (a) Intermittent faults: These faults persist for only a limited period after their initiation. It should be noted, however, that even upon their termination, the system may not behave in the same manner as before the fault initiation.

   (b) Permanent faults: Once occurred, these faults exist forever unless the faulty component is serviced/repaired or replaced by a redundant one, if possible.

2. **Fault severity point of view**

(a) Mono-severity level faults (MSLFs): These are faults that occur only at a single state. For example, a stuck-closed fault in a valve can occur only in one configuration. Other examples include stuck-open fault in valves, floating fault, and hard-over failure (HOF) in electric motors.

(b) Finite multi-severity level faults (FMSLFs): FMSLFs are comprised of a set of MSLFs. Failure of a valve, which may be a two-state failure, being either stuck-open or stuck-closed, is an excellent example of FMSLFs. Other examples include HOF and float failure in actuators.

(c) Infinite severity level faults (ISLFs): This type of faults can take place over a continuum, infinite level of severities. Examples of ISLFs include loss of effectiveness and lock-in-place in electric motors and almost all types of sensor faults, including bias, drift, loss of accuracy, freezing, and sensor calibration error.

2.2 Fault diagnosis and failure prognosis

Fault diagnosis and failure prognosis systems are like a doctor for engineering systems. As doctors do, the fault diagnosis and failure prognosis framework should be able to form a set of knowledge from sensor measurement data. For most cases, unfortunately, measurements do not explicitly represent the states of system health, but they are instead the"symptoms" of the states. Also, measurements are highly likely to be multi-dimensional; i.e., multiple different types of measurements are gathered at the same time, and every measurement is differently affected by various fault events. Moreover, measurements are prone to sensor noises and system disturbances. All of the three factors described above make it difficult to estimate the states of system health correctly [25]. In this regard, tracking every possible fault signature is costly, inefficient, and unnecessary. Instead, specific faults potentially resulting in significant impacts on a system should be analyzed before developing fault diagnosis and failure prognosis framework. FMECA is a useful tool for fault selection

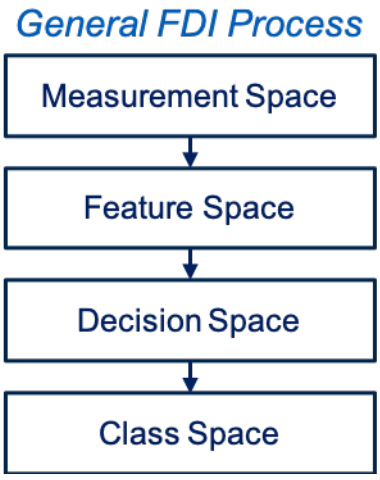

Figure 2.3: General fault diagnosis process (Redrawn based on [26])

analysis [25].

A general process for fault diagnosis is depicted in Figure 2.3 [26]. It starts with obtaining measurements. Measurements, however, do not directly represent the state of health in most cases. Also, measurements are usually high-dimensional, and it makes it challenging to process the signals directly to distinguish one phenomenon to another. Therefore, Measurement data have to be manipulated in intelligent ways such that the manipulated data can represent the states of faults by a fault feature in a reduced dimension. Then, fault and failure modes are identified by the fault features. The results out of this process, which is a process to build knowledge of the system health, should be "enough" and "accurate." In this context, "accurate" means that the results should be robust against uncertainties as well as sensitive to the specific fault of interest.

In terms of the meaning of "enough," on the other hand, medical diagnosis presents an excellent analogy to the depths of required knowledge for system health monitoring. Suppose that a patient sits in front of a medical doctor and talks about symptoms like pains or allergic reactions. The first thing the doctor wants to know is whether the patient has a disease. Once it is determined that the patient has one, the doctor will try to find out what the disease is, where it is, and when it has happened. Then, the doctor will figure out how

severe the disease is and by when it has to be treated before it is too late. All the information described above is the same information that the fault diagnosis and failure prognosis framework provides for engineering systems. A framework first determines whether a system is under the presence of a fault (fault detection.) If a fault is detected, then a framework tracks what the problem is in the system such as where it resides (fault isolation,) and how severe it is (fault identification.) Based on the severity of the fault, a framework predicts a remaining useful life (RUL) of the system before the specific functionality of the system fails due to the evolution of the fault (failure prognosis.)

In this regard, fault detection, isolation, identification, and failure prognosis provide different levels of information. The latter functions, the detailed information about a fault mode. It implies that different levels of a priori knowledge are required for each function. Therefore, there are briefly two different research aspects in the study of fault diagnosis and failure prognosis.

1. Study of fault feature selection and extraction on specific systems

2. Study of signal processing techniques for reliable and robust fault state estimation

This thesis focuses on a general procedure for safety improvement, not for a specific system.

The purpose of fault diagnosis and failure prognosis researches has been focused on developing methods which can accurately detect, isolate, identify faults, and predict failures based on system knowledge by using sensor measurements when a fault occurs during operations. There are two primary evaluation criteria for fault diagnosis and failure.

1. Sensitivity to fault modes

2. Robustness to noises and disturbances

Traditional approaches of fault diagnosis - such as limit checking, frequency spectrum analysis, or fault dictionary approach - were summarized in [27]. Most of these traditional

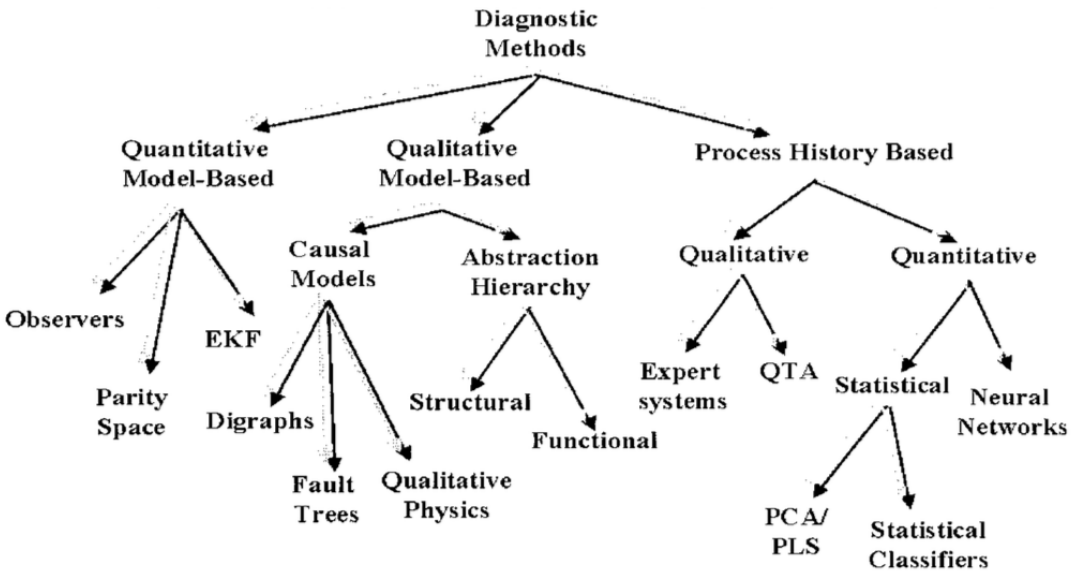

Figure 2.4: Categories of diagnostic methods [26]

approaches were appropriate for simple or static systems whose fault modes and their characteristic behaviors are well-known or whose current states corresponding to measurements only depend on current inputs, not previous states.

For dynamic systems, however, the traditional approaches face significant limitations as addressed in [28]; therefore, many different fault diagnosis methods for dynamic systems have been developed in the past few decades. Venkatasubramanian et al. reviewed categorized FDI methods based on different concepts of approaches as shown in Figure 2.4 [26, 29, 30]. In this review, diagnostic methods are categorized into two different approaches: a model-based approach and process history-based approach. Both approaches were subdivided into quantitative and qualitative approaches, and specific methods under each different branch were reviewed and discussed.

Diagnostic methods described in [26, 29, 30] share two general ideas:

1. Formulating and utilizing a priori knowledge about systems and fault effects

2. Evaluating incoming signals comparing to a priori knowledge formulations by search techniques or learning algorithms

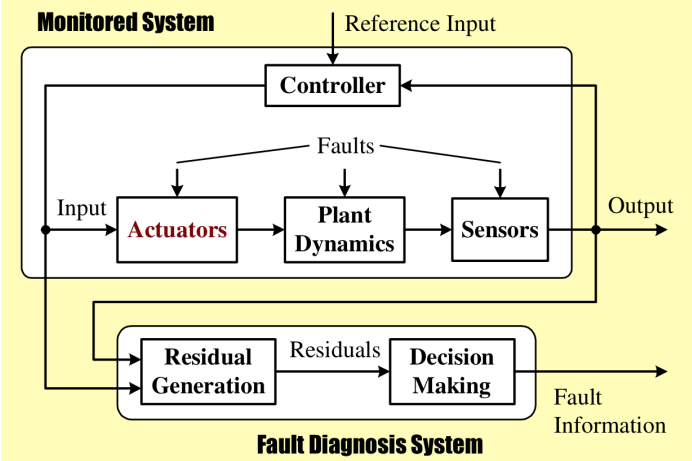

Figure 2.5: A general structure of model-based fault diagnosis framework.

Quantitative model-based methods use mathematical functional relationships between inputs and outputs of systems in order to formulate a priori knowledge. Those input-output relationships are used for generating analytical redundancy. Analytical redundancy is often explained by comparing to hardware redundancy, which utilizes redundant sensor measurements. It requires extra cost and physical spaces for the redundant sensors resulting in limited applicability. On the contrary, analytical redundancy is to make "artificial signals" representing expected system behaviors by system inputs and mathematical functions. Then, artificially generated signals are compared to measured signals, and see if there are inconsistencies: so-called "residuals." Residuals are differences between artificial signals and measured signals, and they are expected to be around zero when the system is healthy and significantly different values when the system changes.

Quantitative model-based approaches using the analytical redundancies were intensively studied in the 1980s and 90s. Patton, Frank, Isermann, Krammer(data?) Figure 2.5 represents a general structure of model-based fault diagnosis using analytical redundancy. As shown in the figure, quantitative model-based approaches are composed of two general steps: residual generation and residual evaluation.

Most quantitative model-based methods start from linear state space models [31] as:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) + R_1 f(t) + Ed(t) \\ y(t) &= Cx(t) + Du(t) + R_2 f(t) \end{aligned} \tag{2.1}$$

where x is a state vector; y is an output vector; u is an input vector; d is an unknown system uncertainty vector; f is a fault vector; R_1 and R_2 are fault effect matrices on system behaviors and output measurements, respectively; A, B, C, and D are system parameter matrices; E is a distribution matrix for d; and t is the time. A basic assumption here is that A, B, C, D, E, R_1, and R_2 are known matrices. Also, a dimension of f is pre-determined, which means that faults of interest are decided.

Lee et al. surveyed the most commonly used diagnostics/prognostics algorithms, their applications, advantages, and disadvantages [32]. Formulation and evaluation methods characterize fault diagnostics methods differently. Usually, there is no single generic method or solution. Instead, different fault models and applications require specific properties of diagnostics methods.

2.2.1 State-of-the-art techniques

A filtering-based approach is widely used in fault diagnosis and failure prognosis applications when fault dynamics is well-known. The filtering-based method uses a fault dynamics model as well as fault-related data to estimate the state of the fault and predict system RUL as depicted in Figure 2.6. It effectively reduces measurement noises and increases the accuracy of the fault state estimation. Also, their inherent property of uncertainty quantification provides a statistical measure of detection confidence. Figure 2.7 illustrates a notional fault feature representation by probability density function (pdf). As the estimate pdf deviates from the baseline, there is a high chance of the presence of a fault. Type-I and Type-II errors impose limits determining the state of health. The most popular filtering-based methods are

Figure 2.6: Filtering-based diagnosis

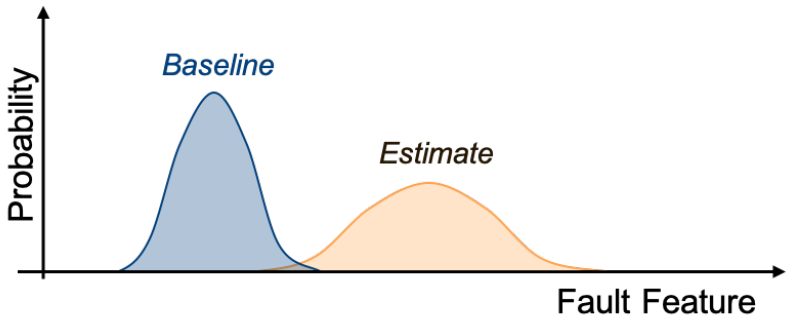

Figure 2.7: Fault detection and identification by probability density function representation

Kalman and particle filtering methods.

The particle filtering-based approach has a benefit over the Kalman filtering-based method. Unlike the Kalman filtering-based approach, the particle filtering-based method does not assume a Gaussian noise; thus, theoretically, it can handle any uncertainty distributions in the measurement. Equation 2.2 shows a general particle filtering-based fault feature estimation model by nonlinear Bayesian state estimation [33].

$$\begin{aligned} x_d(t+1) &= f_b(x_d(t) + n(t)) \\ x_c(t+1) &= f_t(x_d(t), x_c(t), w(t)) \\ y(t) &= h_t(x_d(t), x_c(t), v(t)) \end{aligned} \tag{2.2}$$

where f_b, f_t and h_t are nonlinear mappings, $x_d(t)$ is a Boolean state associated with the presence of a particular fault mode in the system, $x_c(t)$ is the evolution of the system given those operating conditions, $w(t)$ and $v(t)$ are non-Gaussian distributions for the process

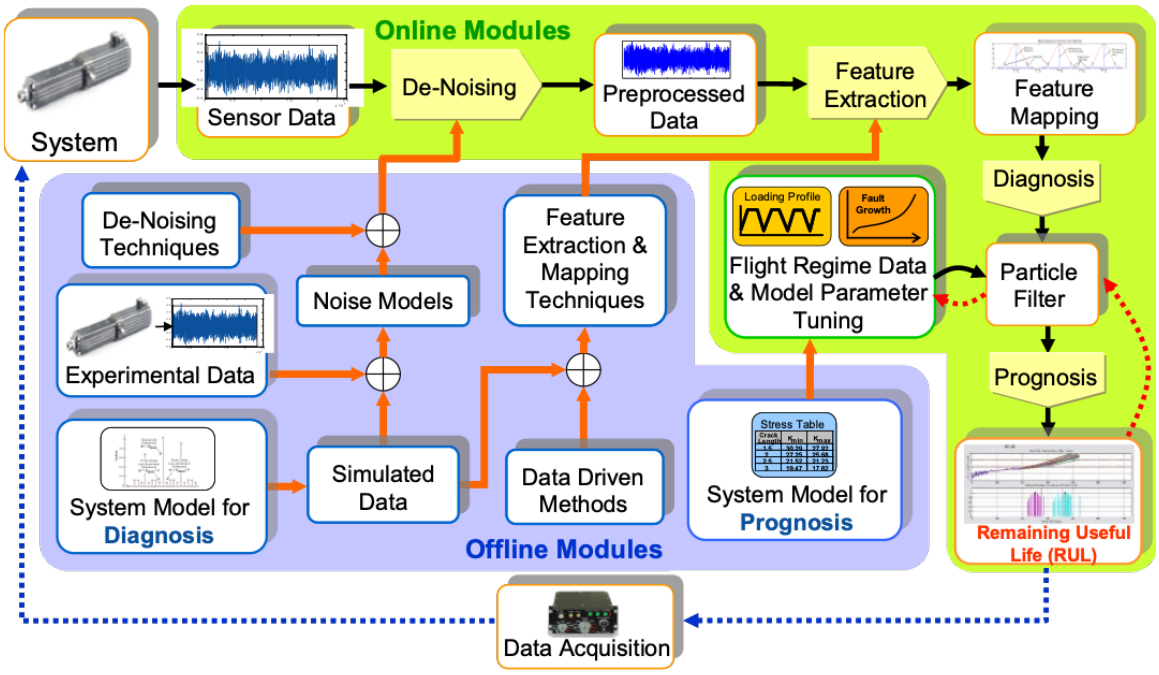

Figure 2.8: Particle filtering-based fault diagnosis and failure prognosis [33]

and feature noises.

Figure 2.8 illustrates a comprehensive online and offline diagnosis and prognosis procedure using the particle filtering method.

2.3 Resilience engineering

The concept of resilience has come to a new design and operational goal for system safety improvement [34]. Historically, for safety management, reliability engineering has majorly focused on preventing fault and failure events and managing them during its lifetime.

Recently, however, system engineering experts question the adequacy of reliability engineering and risk mitigation techniques as complex system safety assessment methods [13]. Risk identification and reliability engineering techniques do not adequately cover the environmental and operational uncertainties. It is simply impossible to prevent threats, system failures, and operational hazards at all times in any systems. Many unpleasant accidents have easily proved it during the missions of carefully designed and operated complex systems: the Space Shuttle Columbia [35], unmanned aircraft accidents due to human fac-

tors [36]. Also, traditional risk management techniques mainly rely on past incidents and their investigations, and they utilize failure probabilities obtained from historical data [37]. Diagnostics, which is one of the methods for system health monitoring, is to detect the root causes of faults and failures, not the risk as of their consequences [13]. Past researches on design approaches for the system safety improvement were briefly reviewed in Appendix A.

In this context, there have been needs of a paradigm shift from a passive, reactive and diagnosis-based approach to an active, proactive and prognosis-based approach, which monitors and assesses risks and system survivability [37]. As an answer to the needs, a new concept, so-called resilience engineering, has been introduced. Madni stated that resilience engineering is to develop a proactive engineering framework with the understanding of failures in complex systems, organizational contributors to risk, and human performance driver [38]. It encompasses the measurement of resiliency, decision support for capability-safety trade-offs, and dynamic evaluation & updates of risk models in order to effectively improve the ability of safety investment prioritization.

System resilience is an evolving concept, which was initially introduced in ecological systems by Holling [17]. Resilience was defined:

- **Definition: Resilience**

    1. A measure of the persistence of systems and their ability to absorb change and disturbance and still maintain the same relationship between populations or state variables [17]
    2. The ability to prepare for and adapt to changing conditions and withstand and recover rapidly from disruptions. Resilience includes the ability to withstand and recover from deliberate attacks, accidents, or naturally occurring threats or incidents [39]

A resilient system was viewed as:

- **Definition: Resilient Systems**

    1. A system that can adjust its functions prior to or following changes and disturbances so that it can go on working even after a major mishap or in the presence of continuous stress, mainly by being able to be proactive on safety [40]
    2. A system that is capable of deploying tactical changes, while supported by its built-in robustness, in order to avoid a given set of threats, or restore its mission capability and health levels, if degraded [13]

Based on the definitions, mission capability is significant for resilient systems. Mission capability was defined as [41]:

- **Definition: Mission Capability**
    - A measure of the results of the mission; given the condition of the system during the mission

System resilience is one of the system characteristics, which contributes to system safety, robustness, and survivability [13]. In this notion of system resilience, failure means the failure of appropriate adaptation to operational disturbances and unexpected events with finite resources and reaction time [40]. Success, likewise, means successful adaptation to the risks to avoid possible dangerous outcomes.

The concept of resilience engineering can be considered as an advanced version of traditional safety and survivability engineering disciplines. Particularly, Hollnagel described challenges that resilience engineering should be able to tackle [40]:

1. Performance conditions are always underspecified.
2. Adverse events can be attributed to an unexpected combination of normal performance variability.

3. Safety management cannot be based on hindsight nor solely rely on error tabulation and failure probability calculations.

A significant difference between traditional safety management approaches and resilience engineering is whether a system can *proactively* reduce system susceptibility or not [40]. The term, proactive, is an essential feature because it means a system can actively remove dangerous environment and prevent unwanted events during the operations. This feature is a new vision of survivability engineering in that traditional safety management approaches have been reactive or passive, and they also focused on vulnerability reduction. With the consideration of proactive characteristics, Hollnagel proposed basic functionality that a resilient system has to be able to: anticipate, monitor, and respond [40]. Figure 2.9 describes the three basic functions of a resilient system and their relationships [13].

1. **Anticipate**: predict and prepare for disturbances, including any changes in operating conditions such as potential threats, operational disruptions, or destabilizing conditions.

2. **Monitor**: monitor risks and threat effects as a part of system performances in addition to mission performance and system health states.

3. **Respond**: react against monitored risks and threats. This functionality is a system property representing flexibility and adaptability.

2.3.1 State-of-the-art techniques

For safety improvement, resilience design approaches have proposed various assessment metrics. Richard et al. suggested a survivability assessment method for aerospace and mechanical systems, inspired by the concept of resilience. It is an epoch analysis regarding the system's dynamic performance response, as shown in Figure 2.10 [42].

Richard et al. proposed two survivability metrics [42]:

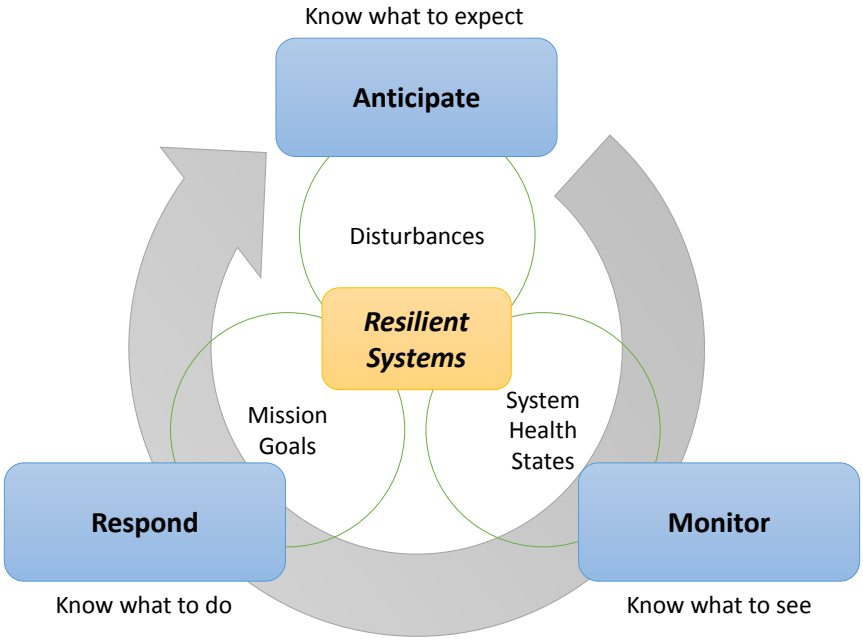

Figure 2.9: Fundamental functionality of resilient systems [13, 40]

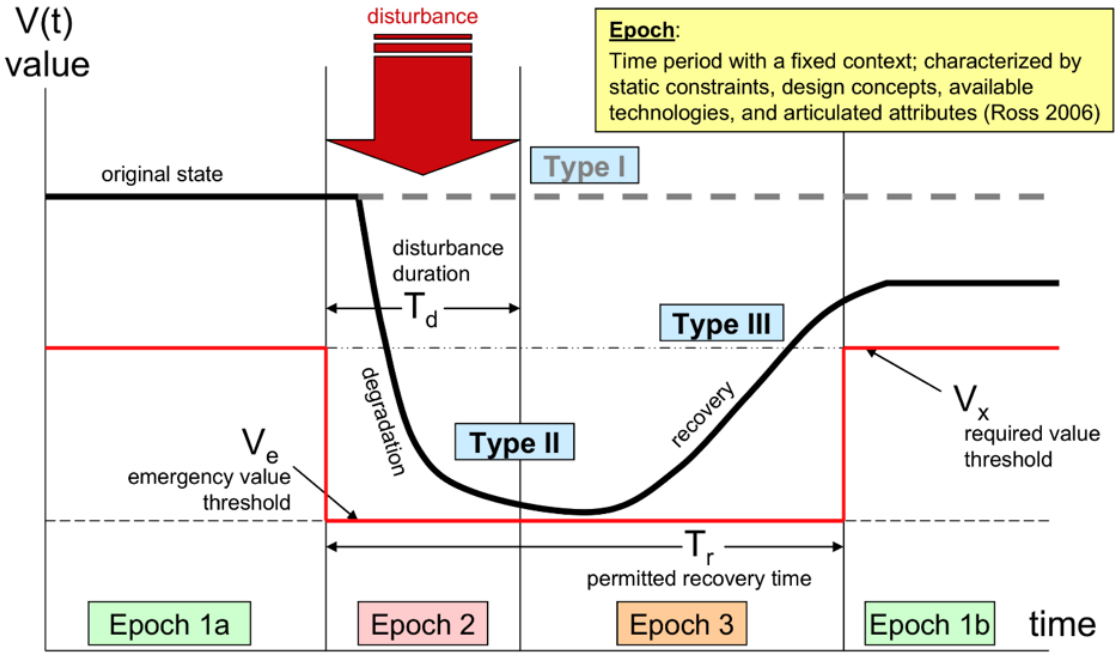

Figure 2.10: Conceptualization of survivability by system performance [42]

1. Time-weighted average utility loss, $\overline{U}_L$, which is defined by

$$\overline{U}_L = U_0 - \overline{U}_t$$

where $\overline{U}_t = \frac{1}{t_{dl}} \int U(t)\, dt$, $U(t)$ is utility function varying over time, and t_{dl} is the design life.

2. Threshold Availability, A_T, which is defined as the ratio of the time that $U(t)$ is above an operable utility threshold (TAT, Time Above Threshold) to t_{dl}:

$$A_T = \frac{\text{TAT}}{t_{dl}}$$

Madni and Jackson proposed generic resilience metrics [38]:

1. Time and cost to restore operation
2. Time and cost to restore configuration
3. Time and cost to restore functionality and performance
4. The degree to which pre-disruption state has been restored
5. Potential disruptions avoided
6. Adaptability within time and cost constraints

Vugrin et al. addressed a recovery effort as well as an impact of disturbance for resilience assessment as depicted in Figure 2.11 [43].

Balchanos reviewed resilience metrics and proposed a resilience assessment methodology in terms of system capability over mission time, namely Epoch, as shown in Figure 2.12. Balchanos suggested resilience metrics based on the concept of Richards method but came up with more metrics for each epoch in terms of absorbability, recoverability, and survivability [13].

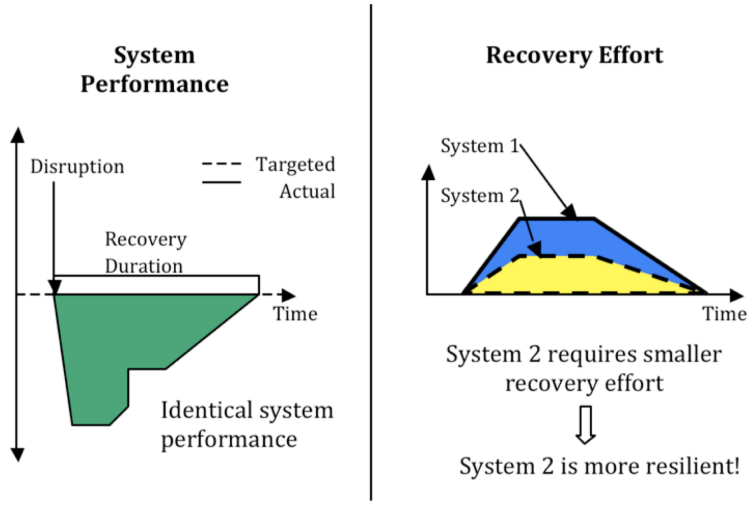

Figure 2.11: Graphical comparison of the resilience of two systems [43]

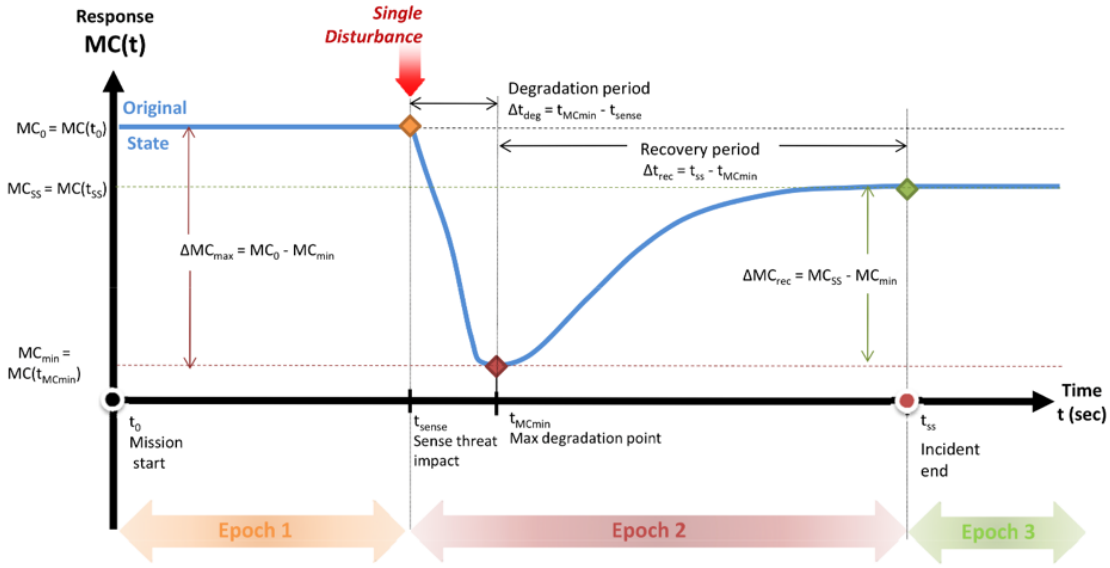

Figure 2.12: Fundamental functionality of resilient systems [13]

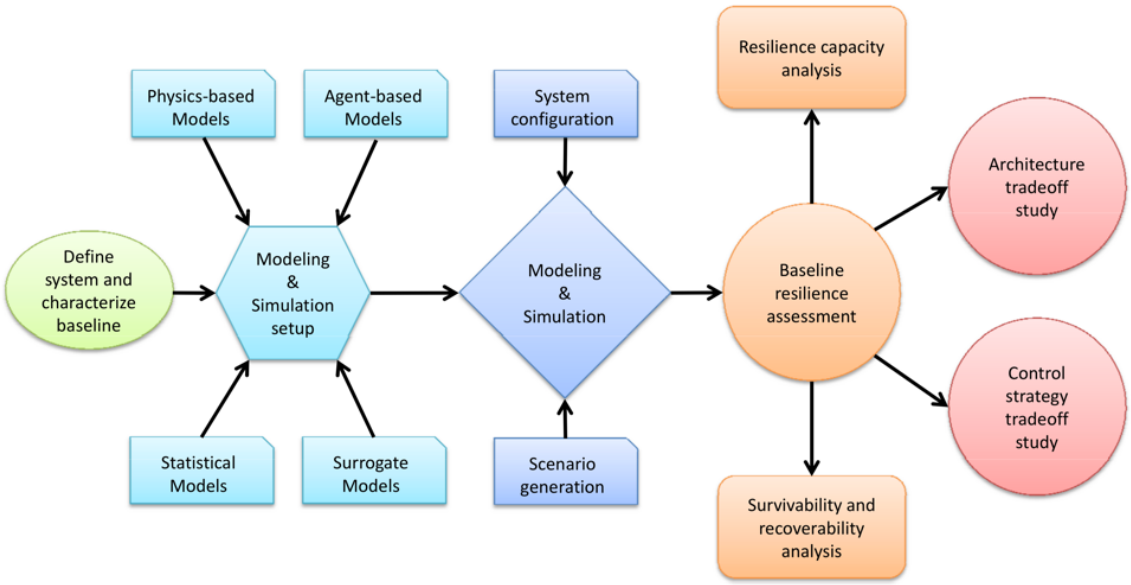

Figure 2.13: TIRESIAS for system resilience assessment [13]

Balchanos' proposed resilience assessment framework is called TIRESIAS: Topological Investigation for Resilient and Effective Systems through Increased Architectural Survivability. The overview of TIRESIAS is depicted in Figure 2.13.

Watson et al. proposed a conceptual framework for developing resilience metrics for electric, oil, and gas infrastructure. He suggested resilience metrics in terms of various factors. Key factors are [34]:

1. Threat: Metrics should show what types of severe disturbances and how severe disturbances should be considered.

2. Performance: Metrics should be able to indicate how well a system satisfies given mission goals. It is not about system characteristics such as redundancies; instead, it measures how well a system delivers its performance in the presence of threats.

3. Consequence: Metrics should represent the ultimate consequences of severe disturbances. Sometimes, performance-based metrics may be measurable quantities of consequences.

Figure 2.14: Overview of Resilience Analysis Process [34]

4. Uncertainty: By the definition of resilience, a system needs to be able to handle unknown and unanticipated disturbances at a certain level.

Based on the resilience key factors, Resilience Analysis Process (RAP) was introduced as illustrated in Figure 2.14.

2.3.2 Research gap analysis

In summary, as an emerging concept of system safety, resilience engineering is stemmed from reliability and robust engineering. Resilience engineering research proposed resilience assessment metrics. The resilience metrics have been represented by system performance & mission capability degradation, uncertainty impact and recovery time, recovery efforts, consequences, or threat levels in various research. Resilience design approaches proposed a system design framework focusing on resilience metrics improvement.

By the inherent nature of resilience, resilient systems are required to have adaptation capability. The performance of adaptability is vital to resilience improvement, not only system reliability and robustness. Therefore, resilience design approaches evaluate the system

resilience with a certain level of adaptability. Alternatively, the adaptability requirements are derived from the desirability of system resilience.

The adaptation capability is highly relevant to control systems. However, resilience engineering or resilience design approaches do not explicitly address the control problem in terms of system resilience. Research on the actual implementation of a reconfigurable control system having a concept of system resilience improvement will help the realization of resilience design.

Research Gap 1

A resilience design approach does not explicitly address the practical implementation of adaptation/recovery policies under threats.

2.4 Fault Tolerant Control Systems (FTCS)

The socio-technical system in risk management was described by Rasmussen and Svedung [44], and Leveson depicted its hierarchical model as shown in Figure 2.16 [45]. This hierarchical model is a general working procedure when an organization (system) faces undesirable events or a chain of events. Based on a particular root cause, at a certain point of time, a system may encounter a critical event, causing hazard release. In response to those events, different hierarchical supervisors react based on observations and reports. In this regard, Leveson stated that [45]:

> "Safety can be viewed as a control problem, and safety is managed by a control structure embedded in an adaptive socio-technical system. The goal of the control structure is to enforce constraints on system development (including both the development process itself and the resulting system design) and on system operation that result in safe behavior."

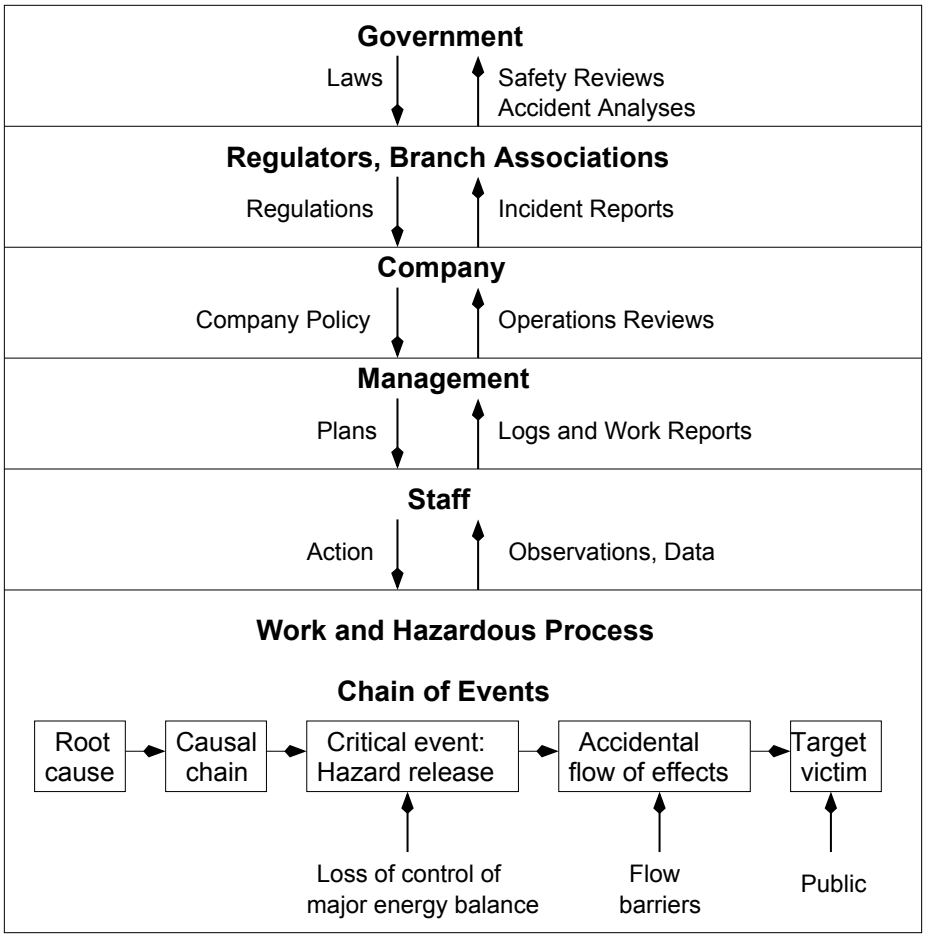

Figure 2.15: Socio-technical model of system operation [45]

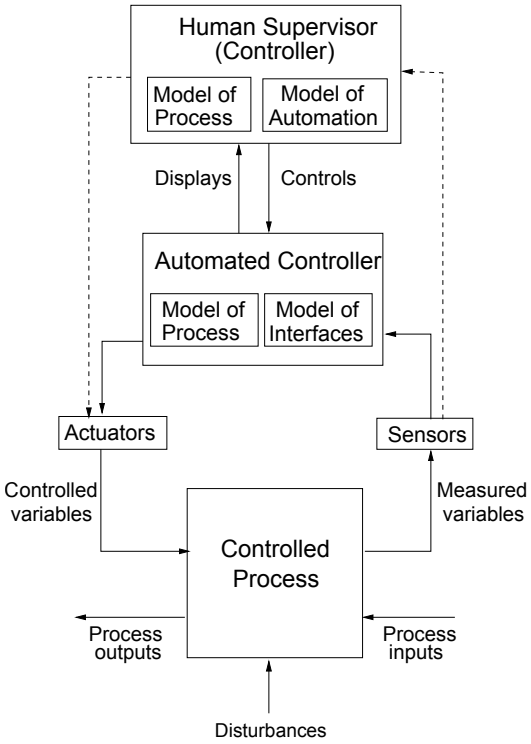

Figure 2.16: A generic control architecture with process modules [45]

It shows that control theory is one of the critical approaches to safety management because accidents happen when a control system does not correctly deal with external disturbances, component failure, or dysfunction among system components [45]. Figure 2.16 shows a typical socio-technical control architecture. There are multiple control loops to handle disturbances; an automated controller operates an inner-loop control, and a human controller supervises an outer- loop control. Both controllers obtain information about system states from feedback measurements and command control variables to make system states of being as close to the desired states as possible. Besides, there can be interactions between the two controllers. Based on what types of displays the inner-loop controller provides, the outer controller reactions will vary. As such, a system will behave differently based on how well a control structure works.

Now, what types of disturbances a control structure can deal with determines the types of safety-related problems that a control structure is designed to solve. Figure 2.17 illustrates types of disturbances and their safety problems [46]. If the disturbances are from

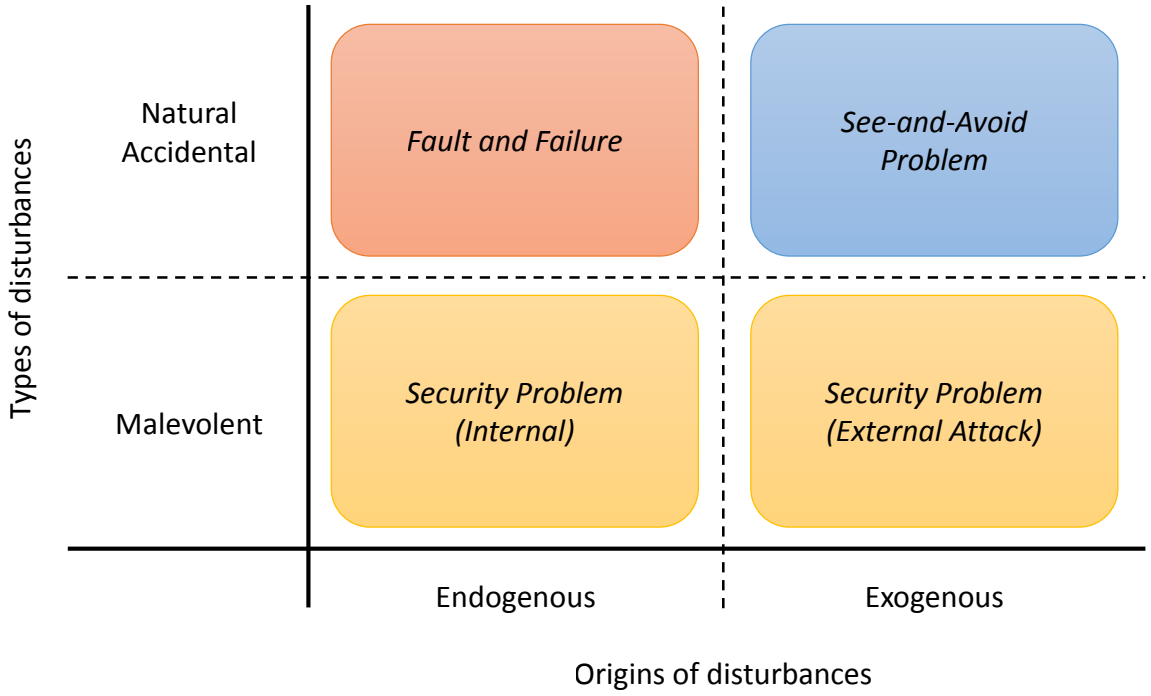

Figure 2.17: Safety problems with respect to types of disturbances [46]

malevolent sources, they are more about security problems wherever they are originated. If they are accidental and originated from exogenous sources, then they are "*See-and-Avoid*" types of problems; e.g., one of the significant safety concerns for utilization of UAS in National Airspace System (NAS) is how to reduce significant mid-air collisions during the operation [47]. Finally, disturbances from natural and endogenous sources are system fault and failure problems.

All problems described above are essential for system safety. However, 'natural and accidental' types of disturbances are more common for any systems, whereas 'malevolent' disturbances are more severe in military systems. Also, just like the same argument of system capability and system dependability, the ability of "See-and-Avoid" heavily depends on the quality of system integrity. It implies that system fault and failure analysis and their treatments will be critical to entire system safety improvement.

For the treatment of fault/failure events, a conventional feedback controller may not suffice in terms of system performances or stability when actuators, sensors, or other system

components malfunction; thus, new control system designs have been studied, specifically for safety-critical systems such as aircraft, spacecraft, nuclear power plants, or chemical plants. [48, 49]. For these safety-critical systems, there has been extensive research in the past several decades as a new controller design, which can tolerate undesirable fault effects and improve system reliability and availability. This type of control systems is called Fault Tolerant Control System, FTCS.

In the historical point of view, a large amount of research on FTCS was driven by system designs of aircraft flight controls [50]. The research was motivated by commercial aircraft accidents [49]. Delta Flight 1080 encountered the jammed elevator during its flight [51]. Fortunately, in this accident, the pilot saved the aircraft with his knowledge about actuation redundancies and with his experiences in the control reconfiguration. However, most of the failures caused severe damages and casualties such as the American Airlines DC-10 (Flight 191) crash in 1979 [52] and the EL AL Boeing 747-200F (Flight 1862) freighter in 1992 [53]. To avoid those fault-related flight accidents, control systems that aid pilots by providing proper accommodation started to gain the spotlight. Recently, other safety-critical industries have also picked up the idea due to increased safety and reliability demands [49].

FTCS is briefly categorized into two sub-categories: Passive FTCS (PFTCS) and Active FTCS (AFTCS) [49]. PFTCS aims at designing control systems to be highly robust so that control systems can deal with pre-designated unwanted situations without any changes of control systems [54]. AFTCS, on the contrary, intends to design control systems to enable control systems to react to undesirable events by reconfiguring control actions flexibly. In this type of control systems, the controller compensates for fault effects by pre-determined control laws [55, 56, 57] or online control law adaptation [58, 59]. Figure 2.18 summarizes general comparisons between PFTCS and AFTCS.

Since PFTCS does not require the online adaptation, the way that a controller works is as simple as the traditional controllers do. It uses fixed control laws and does not require

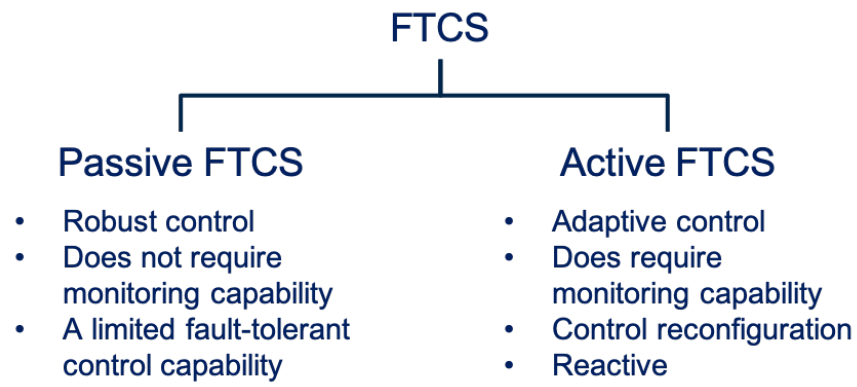

Figure 2.18: A conceptual overview of AFTCS [49]

online fault detection and identification functionality [49, 60]. However, since PFTCS is supposed to be designed to cope with every possible operating condition, it does not provide an optimal control law for the standard condition as well as all possible faulty conditions [60].

On the contrary, AFTCS is a control system that flexibly reacts to undesirable fault & failure events in the middle of missions automatically [49]. Since it is adaptive, it inevitably requires online monitoring capability in order to recognize how the system works and whether the system is faulty or not. It requires expensive computations on the fly real-time. As computational power and the size of central processing units (CPUs) have been considerably improving, AFTCS is a very appealing approach for system safety improvement. In this paper, AFTCS is the research area of interest.

AFTCS is generally composed of necessary modules as listed below and shown in Figure 2.19

1. Fault Detection and Diagnosis (FDD)

2. Reconfiguration mechanism

3. Reconfigurable controller

4. Command/reference governor

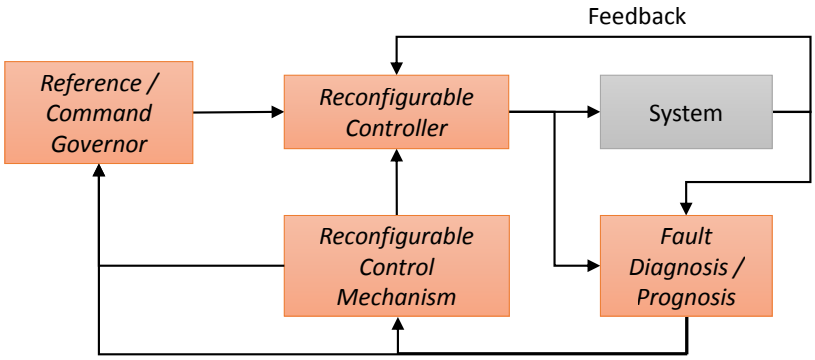

Figure 2.19: Overview of an active fault tolerant control system structure [49]

It has closed feedback loops coming from sensors onboard just like traditional control systems. Measurements are fed into navigation modules estimating the current states of a system. By the guidance module, references, for example, target positions that a system has to be headed, go into a controller module. As AFTCS, the fault detection module monitors fault modes online. Once it detects a fault, the reconfiguration mechanism finds adaptive control actions based on its strategy.

The inclusion of Fault Detection and Diagnosis (FDD) module distinguishes AFTCS from PFTCS [49]. FDD monitors possible faults in critical sub-systems or components. Then, based on the fault-related knowledge from the first module, the second module, reconfigurable mechanism, optimizes a control strategy such that a system can maintain stability and system performances under faulty conditions as close as possible to the original system performances. The third module, reconfigurable controller, reconfigures controller algorithms as decided in the reconfiguration mechanism module. The last module, command/reference governor, reshapes input references when actuators are saturated. Saturation of actuators usually makes it difficult for the system to follow control inputs and directed waypoints/paths. All four modules are necessary for the success of AFTCS.

Zhang et al. reviewed literature about reconfigurable fault-tolerant control systems [49]. It classified reconfigurable control design methods into mathematical design tools, controller design approaches, reconfiguration mechanisms, and applied system characteristics for typical AFTCS designs as shown in Figure 2.20. The literature reviewed in this survey

uses the fault detection and diagnostics information for reactive capability.

Figure 2.21 illustrates how AFTCS is different from nominal control or PFTCS in general. The left figure depicts a designed operating point a conventional feedback control system. For PFTCS, a designed operating point is chosen away from the nominal operating point by considering possible drifts due to fault modes as shown in the center figure. It may imply that a system mainly operates on a sub-optimal operating point because the rate of the fault or failure is usually not much high. Unlikely, AFTCS allows a system to operate on an optimal design point when the system is healthy. When a fault occurs, AFTCS redistributes control authority to put an operating point back to the design point as close as possible.

2.4.1 State-of-the-art Techniques

Prognostics-based reconfigurable control methods partly address the long-term system utilization problem in the presence of fault or failure modes. The prognostics routine provides RUL. Then, a reconfigurable controller optimizes its control strategy to extend the RUL so that the RUL exceeds the mission time required.

Under small disturbances, the low-level reconfiguration module compensates for the effects of small disturbances by adjusting set points for the actuator components. In the case of extreme disturbances, however, due to the severe degradation of the system capability, the system is not able to satisfy mission requirements; thus, it cannot perform/complete the given mission. In order to resolve this issue, the middle-level reconfiguration module is introduced to extend and recover the system capability by reconfiguring the guidance and control strategy. The system capability recovery in terms of an extended RUL is achieved at the expense of degraded system performance. It is noted that the proposed reconfiguration strategy does not entail any hardware changes (replacement or insertion of new hardware). It addresses only the software components of the framework. Therefore, the trade-off between the performance requirements and increased RUL must be carried out correctly in

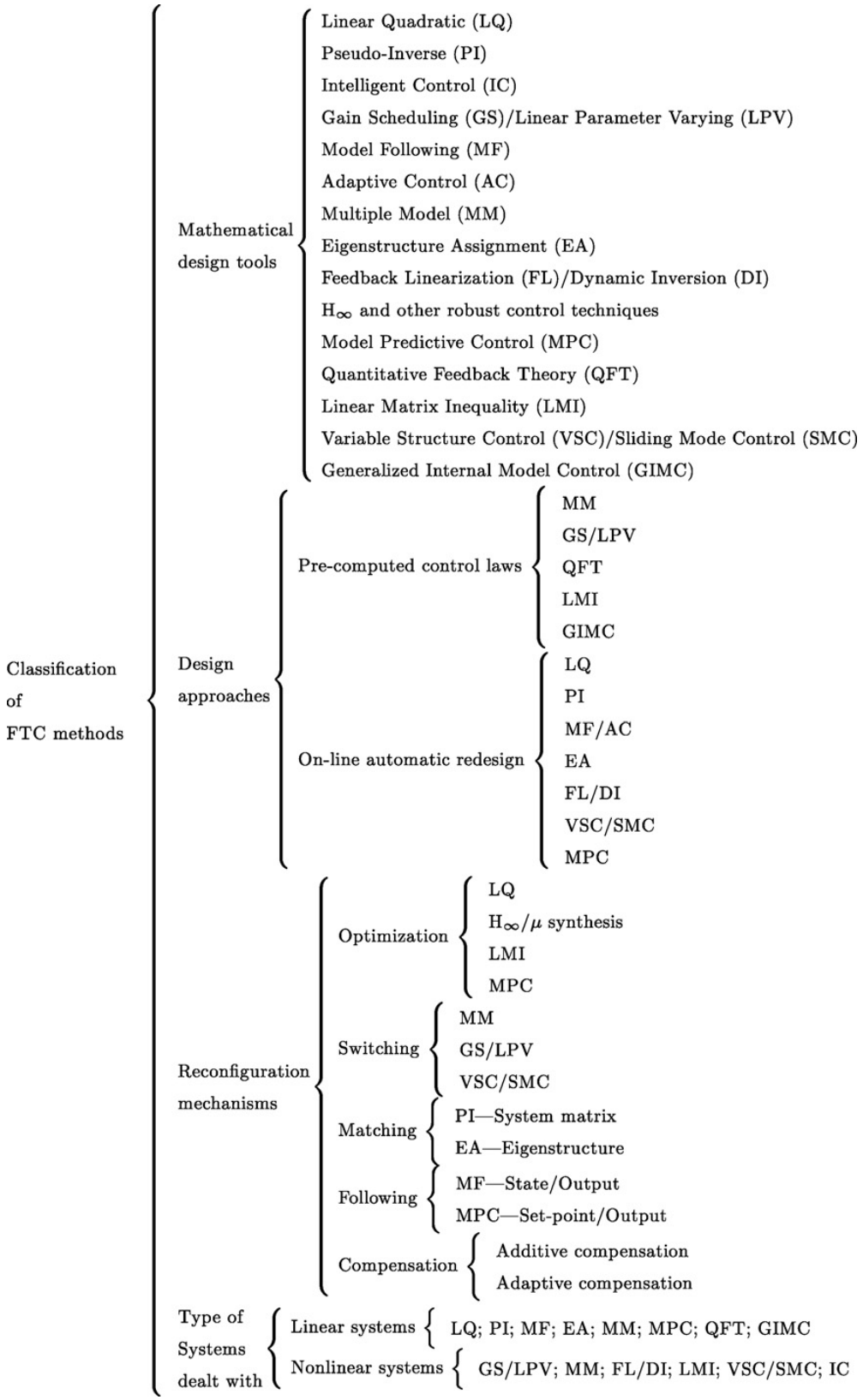

Figure 2.20: Classification of AFTCS [49].

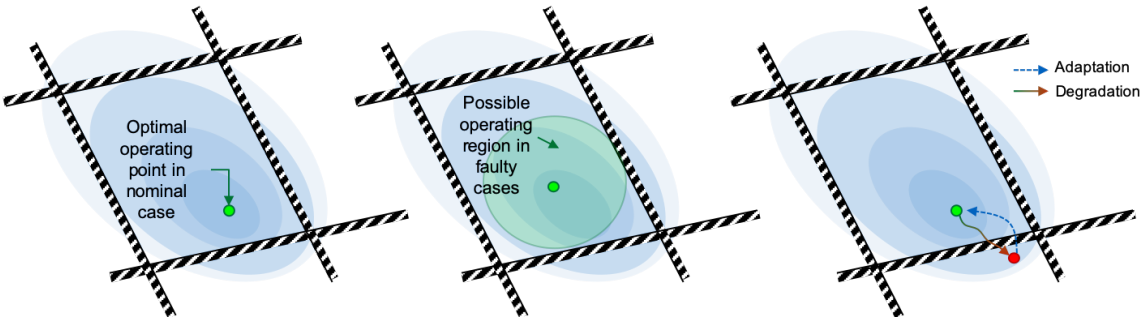

Figure 2.21: A conceptual overview of control systems

the middle-level reconfiguration module. This trade-off is the essence of the middle-level reconfiguration.

In a broad perspective of AFTCS, Clements proposed a concept of hierarchical control architecture [61]. He showed the interconnections among fault detection & identification, set-point controller, control redistribution, control gain adaptation, and component restructuring in three hierarchical levels as depicted in Figure 2.22.

Drozeski developed a three-tier hierarchical control scheme based on Clements' control architecture. Drozeski formulated a general form of an objective function for high-level AFTCS as expressed in Eq. 2.3 [60].

$$J\left(M,R\right)=U\left(P_e,M,M_{com}\right) \tag{2.3}$$

where U is a cost function that quantifies the usefulness of the vehicle for its mission. P_e is a measure of the closed loop performance of the vehicle. It is a function of a fault mode, F_m, and restructuring & reconfiguration actions, R. M_{com} is an assigned mission, and M is a set of parameters for mission adaptation such as nominal jerks, accelerations, and velocities used by path planners. Drozeski used parameter-based FDI and system identification routine to estimate system and fault states. As a reconfigurable controller, an adaptive neural network flight controller by dynamic inversion and pseudo control hedging method were employed. In the test case, mono severity-level faults were considered.

Ge et al. introduced a higher level of adaptive system framework by using an Au-

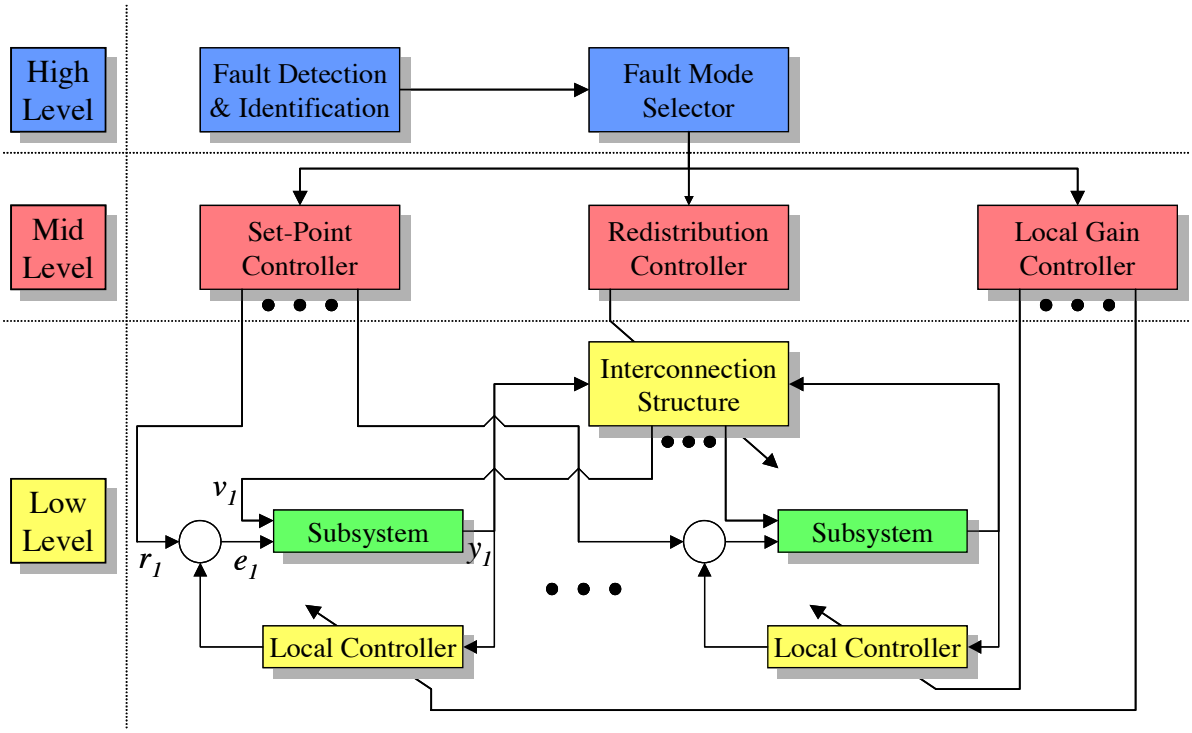

Figure 2.22: Clements hierarchical fault-tolerant control architecture [61])

tomated Contingency Management (ACM) concept [62]. The proposed framework is a conceptual structure, which broadened the perspective of FTC, from strategic planning to tactical and trajectory re-planning as shown in Figure 2.23. Technical methods for each planning function were not addressed.

Tang et al. extended the ACM framework by integrating a prognostics module [63]. Figure 2.24 depicts a conceptual ACM+P system hierarchy. In the test cases, a swashplate collective actuator failure mode was tested as infinite severity-level fault behavior. The proposed framework drove the faulty swashplate of a helicopter so that the faulty component failed at a minimal impact position.

Later, Tang et al. tested Prognostics and Health Management (PHM) enhanced ACM framework with a real-time robot test application [64]. As the battery life for a degrading property, a path of unmanned ground vehicle was optimized by the D* algorithm considering mission time, easiness of travel, and RUL. This study addressed the change of mission time for the optimization criterion.

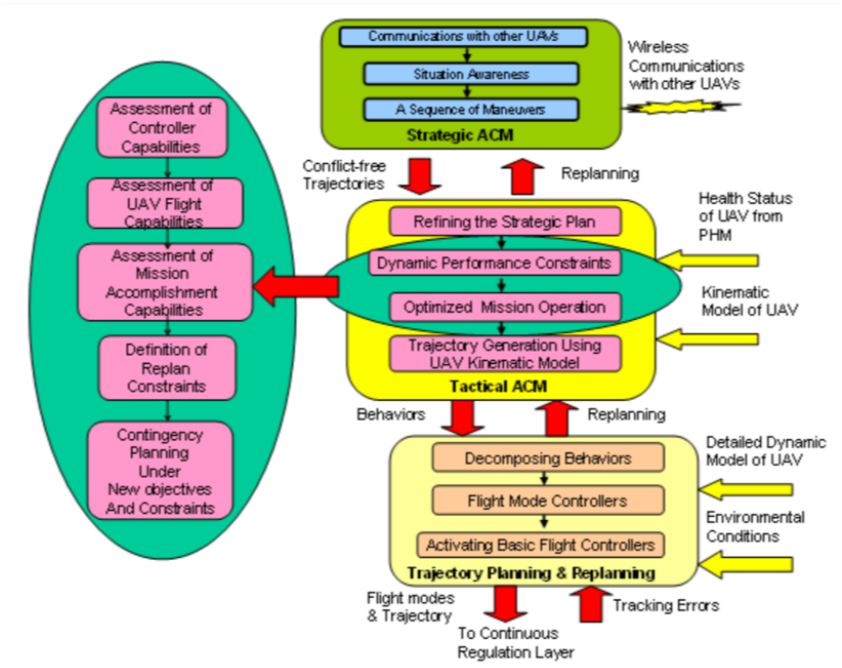

Figure 2.23: Overview of Automated Contingency Management framework [62]

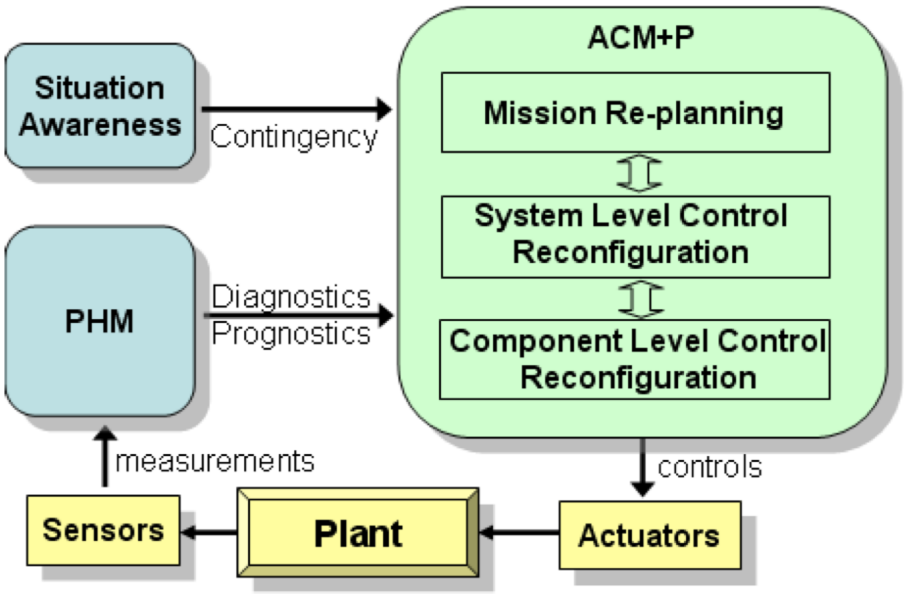

Figure 2.24: Conceptual ACM+P system hierarchy [63]

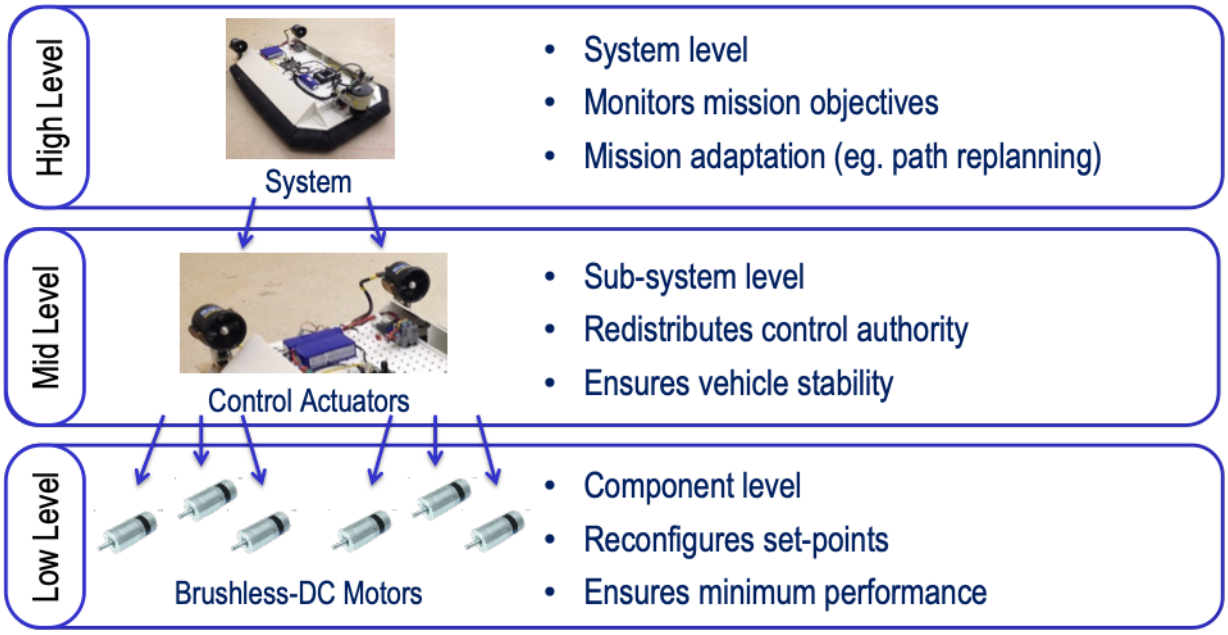

Figure 2.25: Reconfigurable control architecture with three tier strategies (modified and redrawn based on [65])

In the meantime, Brown et al. introduced a 3-tier hierarchical strategy for the reconfigurable control architecture [65] as illustrated in Figure 2.25. The 3-tier hierarchical architecture presented three different reconfiguration problems through the levels of impact of a fault mode. With a minor impact, the low-level reconfiguration adjusts control set-points to minimize the fault growth allowing deviations from the original references. As severer fault modes degrade system health, mid or high-level redistributes control authority or adapts its mission. In this study, he proposed the low-level control reconfiguration by MPC as a reconfigurable controller and the particle filtering-based fault diagnosis and failure prognosis routine. At each control prediction in MPC, set-points were reconfigured so that RUL within the MPC time window exceeded RUL required.

Bole proposed prognostics-enhanced load allocation framework [66]. He addressed the challenges in predicting RUL due to the changes in fault growth rate as loading conditions vary. Just like [65], MPC and a particle filtering-based fault diagnosis and failure prognosis routine were used. Figure 2.26 illustrates the impact of loading conditions on RUL [66].

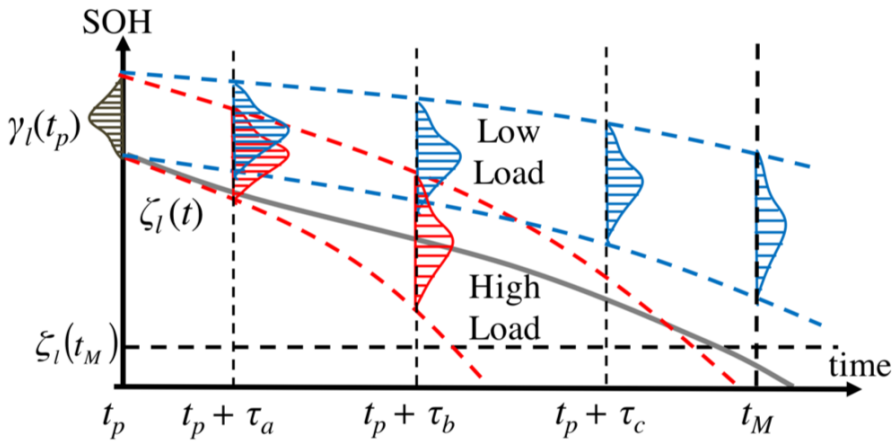

Figure 2.26: Evaluation of prognostic constraint at several finite horizons for two sample component loading profiles denoted 'High' and 'Low' [66]

2.4.2 Research gap analysis

In summary, conventional AFTCS recover the operating point as close to the design point as possible when a fault, specifically an actuator fault mode, occurs during the operation. However, conventional AFTCS do not address the impact of a fault on long-term system performances.

Prognostics-enhanced reconfigurable control methods address challenges in the loss of system usable time due to fault modes as well as the operating point degradation. A prognostics routine produces RUL prediction, and a reconfigurable controller modifies the control strategy to extend the predicted RUL so that the RUL exceed a given mission time required. In case that a fault growth rate is loading conditions-dependent, MPC-based reconfiguration helps regulate control inputs so that the State-Of-Health (SOH) prediction of the MPC optimal control time window resides within acceptable SOH bounds.

In this regard, the prognostics-enhanced reconfigurable control methods assume a mission terminal time is known. The mission time required, however, may change as a control strategy is reconfigured in the presence of a critical fault mode, which makes system performances significantly impaired. For instance, if a control reconfiguration causes a slow movement of the faulty unmanned vehicle, a mission time to destination inevitably in-

creases. It means that both RUL and the mission time required increase. In this example, therefore, time is not a proper mission capability domain.

Research Gap 2

Prognostics-enhanced reconfigurable control methods assume that a complete mission time is given. $\rightarrow$ Comprehensive long-term mission capability has not been addressed.

Likewise, acceptable SOH bounds are highly dependent on a mission time required and mission profile in MPC-based reconfigurable control methods. It implies that improper SOH bounds could cause unsatisfactory adaptation and catastrophic failure.

Research Gap 3

Mission capability trade-off has not been addressed.

2.5 Problem Definition and Research Objectives

Research gaps identified in the previous section show the research opportunities and requirements. The research gaps are summarized below:

Research Gap 1

A resilience design approach does not explicitly address the practical implementation of adaptation/recovery policies under threats.

Research Gap 2

Prognostics-enhanced reconfigurable control methods assume that a complete mission time is given. $\rightarrow$ Comprehensive long-term mission capability has not been addressed.

Research Gap 3

Mission capability trade-off has not been addressed.

Research Gap 1 demonstrates the need for a reconfigurable controller, which is suitable for a resilient system. For a resilient system, an adaptive controller needs to reconfigure control strategy considering long-term mission capability as well as stability and system performance recovery. Research Gap 2 and 3 indicates that the current reconfigurable control methods cannot deliver the performance requirements as a control system for resilient systems.

In this regard, the research problem of this study is defined below:

Research Problem Definition

A reconfigurable control method, which comprehensively helps meet long-term mission requirements in the presence of a critical fault mode, is missing.

This research hypothesizes that the long-term mission capability prediction is required for a proper control reconfiguration considering system resilience improvement. The prediction of the long-term mission capability must address uncertainties in fault growth patterns and rates.

In order to fill the research gaps, this research aims at developing a design methodology for a reconfigurable controller, which can handle critical fault modes, so that the faulty system still be able to satisfy the mission requirements without a catastrophic failure. The

proposed reconfigurable control methodology will help improve system resilience. The over-arching hypothesis and research objectives are summarized below:

Over-arching Hypothesis

If long-term mission capabilities are considered in control reconfiguration in the presence of a critical fault mode, system resilience will be improved.

Research Objectives

To develop a design methodology for a reconfigurable controller, handling critical fault modes in consideration of system resilience improvement

The next chapter describes the proposed technical approach for the resilience-enhanced reconfigurable control framework. It reviews possible alternative methods, and justifies the selection of proper methods by deriving a set of research questions. In order to prove the efficacy of the proposed methods, hypotheses were raised accordingly.

CHAPTER 3
TECHNICAL APPROACH

This chapter introduces a novel Resilience-Enhanced Reconfigurable Control framework for the resilient design of complex engineering systems. Figure 3.1 depicts the main modules of the resilience-enhanced reconfigurable control framework. The Case-Based Reasoning (CBR) module detects and identifies a faulty system condition during operations and proposes a reconfigurable control strategy to the long-term mission capability prediction module. The long-term mission capability prediction module determines an adaptation parameter and updates it online, considering consequences. Finally, a Model Predictive Control-Differential Dynamic Programming (MPC-DDP) module redistributes the control authority in order to maintain acceptable stability bounds while performing a short-term performance trade-off.

The proposed framework is configured in terms of feedback loops, as shown in Figure 3.2. It continually monitors the system states, adjusts the level of reconfiguration if needed, and feeds adapted control inputs to the system in the presence of faulty conditions.

The major difference of the proposed framework from the past reconfigurable control methods is the consideration of system resilience within the picture of control reconfiguration when the system operates under the presence of critical fault modes. In this regard, each module needs to address the fundamental properties that resilient systems should have adequately. Table 3.1 summarizes a question set that each module will have to answer.

Module 1 focuses on stability and reference-tracking performance recovery as an AFTCS. Failing to maintain the system stability and reference-tracking performance could cause catastrophic system/mission failure quickly. With the knowledge of the fault presence and its level of degradation, DDP optimizes the sequence of control inputs for a finite time window. MPC extends the usability of DDP to a longer time than the size of the finite time

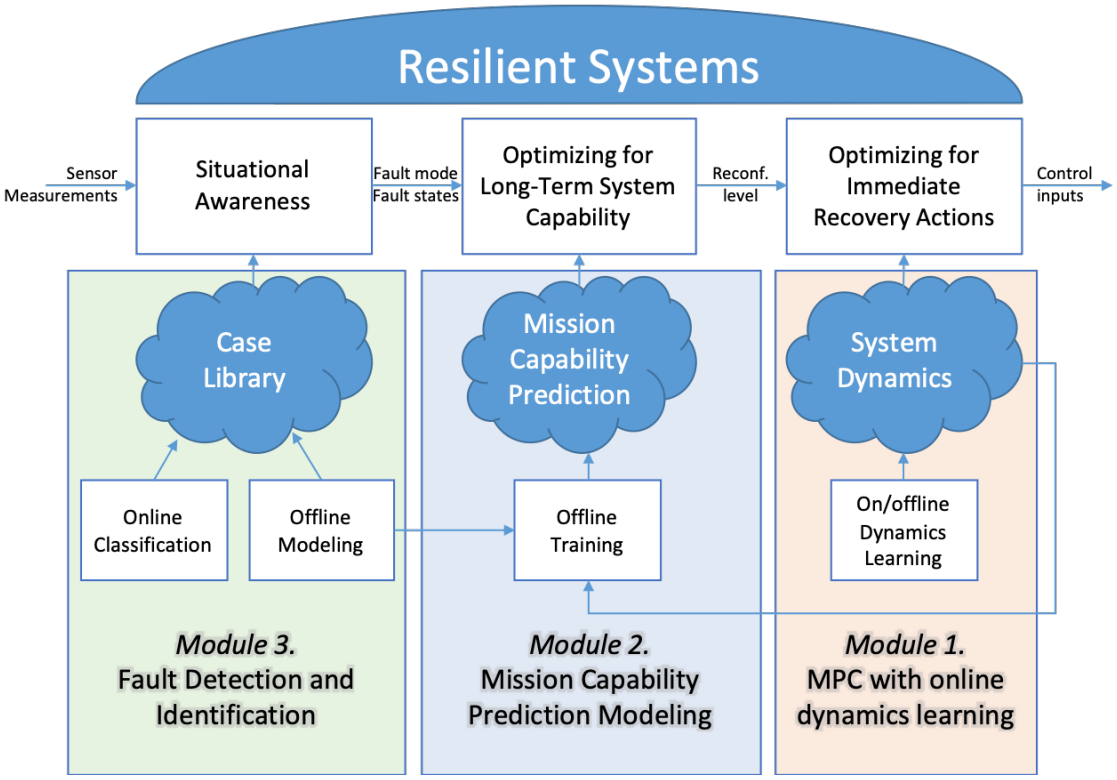

Figure 3.1: Resilience-enhanced reconfigurable control framework with three fundamental modules: MPC-DDP, longterm mission capability prediction, and fault detection and identification

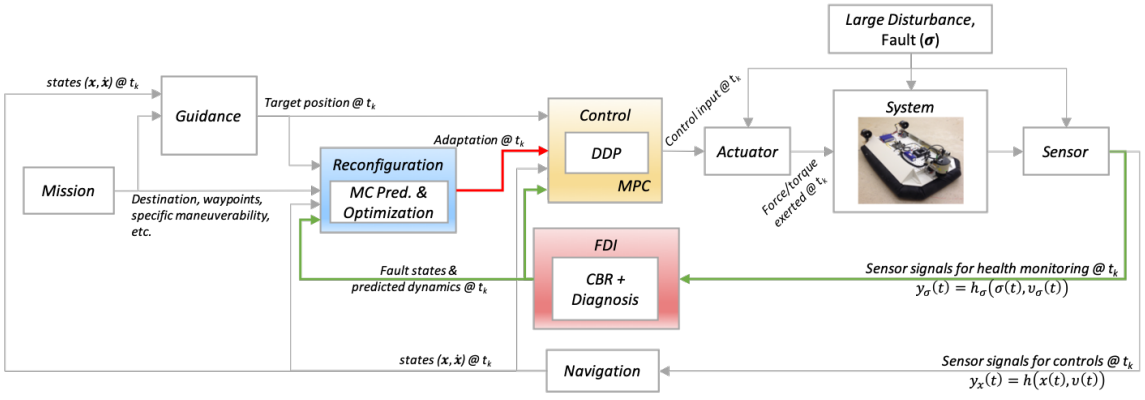

Figure 3.2: Overview of the proposed framework in a closed loop control

Table 3.1: A question set to be addressed by the proposed framework

Module	Question Set
Immediate Recovery (Module 1)	Will the stability be maintained? What is an optimal trajectory? What are the control inputs for proper tracking performances?
Long-term Recovery (Module 2)	How much will the faulty system be able to operate once a fault occurs? How aggressive should the faulty system be controlled?
Situation Awareness (Module 3)	Does a fault occur? What is the fault mode? How severe the fault? How fast will the fault grow?

window of DDP.

Now, the question is if the design operating point would be a desired operating point under the presence of a (critical) fault mode. A faulty system is not the same as its design condition. Maintaining operating points close to the design operating point could expedite the growth of the fault. If it does, the design operating point could be out of the feasible area under a faulty condition considering mission requirements. It implies that a new desired operating point has to be defined in order to prevent the faulty system from mission and system failure if necessary.

To help the understanding, Figure 3.3 depicts the idea of the impact of a critical fault mode to notional constraints and desired operating points. The blue-shaded ellipses represent a notional shape of the objective function. The objective function is usually defined by system-performances, control efforts, and their corresponding coefficients. The left figure shows the impact of a critical fault on the operating point. The operating point is shifted from the designed point to the outside of the feasible region. The right figure represents a desirable adaptation with a new objective function and constraints involved.

In this regard, this study was mainly motivated by the question about how to determine

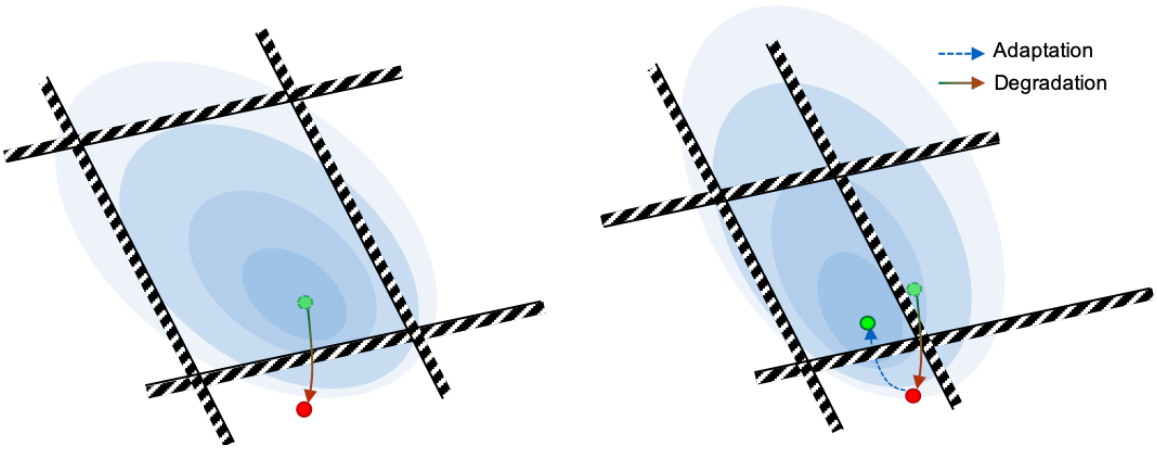

Figure 3.3: Notional impact of a critical fault and desirable reconfiguration. The left figure shows the impact on the operating point. The right figure represents desirable adaptation with a new objective function.

the objective function so that the faulty system could successfully complete its mission considering long-term mission capability, not only the immediate performance recovery.

Module 2 in the proposed framework determines the shape of an MPC-DDP objective function by adjusting an adaptation parameter in MPC-DDP formulation once a critical fault mode is detected. For that, a simulation-based mission capability modeling approach was introduced.

The simulation-based mission capability modeling and optimization approach is inevitably highly sensitive to fault growth patterns and rates. However, there is no guarantee that the same fault mode grows the same way every time. In order to address the issue, a Case-Based Reasoning (CBR) approach was introduced in Module 3. Module 3 identifies a fault growth model as well as detects and identifies a fault mode. Long-term mission capability prediction is then reasoned by experiences (cases) in CBR, and tossed it to Module 2 for the adaptation parameter optimization.

So far, Chapter 3 described the overview of the proposed technical methods module-by-module. Next section explains how the proposed methods have been chosen by literature reviews, detail descriptions of alternative methods, and their comparisons. In order to help the justification, a set of research questions have been raised. Then, hypotheses and

experiment plans have been followed in order to justify that the proposed methods and framework are right ways of system resilience improvement.

3.1 Overview of the proposed technical methods

3.1.1 Module 1: Immediate Recovery by MPC-DDP

A cost function for optimal control is written as:

$$V\left(x\left(t_0\right),t_0\right)=min_u\left[\int_{t_0}^{t_0+T} l\left(x\left(\tau\right),u\left(\tau\right),\tau\right)d\tau+\Phi\left(x\left(t_0+T\right),t_0+T\right)\right] \tag{3.1}$$

where V is a value function, x is a system state vector, u is a control input vector, t_0 is a control time, T is a time window, l is a running cost, and Φ is a terminal cost.

The system dynamics impose an equality condition, while physical constraints are introduced as inequality conditions.

$$\begin{aligned}\frac{dx}{dt}&=F\left(x\left(t\right),x_f\left(t\right),u\left(t\right)\right)\\ \frac{dx_f}{dt}&=F_f\left(x_f\left(t\right),s_f\left(t\right)\right)\\ s_f\left(t\right)&=F_s\left(u\left(t\right),t\right)\\ g\left(x\left(t\right),u\left(t\right)\right)&\leq 0\end{aligned} \tag{3.2}$$

where F is a system dynamics, x_f is a fault (feature) state, s_f is a stressor for the fault growth, F_f is a fault growth dynamics, F_s is a stressor-control input mapping function, and g is a state and input constraints.

The running cost function, l, is typically formulated as a quadratic function, as shown

in Eq. 3.3. It is a linear combination of two terms: system performance and control effort.

$$l\left(x\left(t\right),u\left(t\right),t\right)=\frac{1}{2}\left(x\left(t\right)-r\left(t\right)\right)^{T}K\left(x\left(t\right)-r\left(t\right)\right)+\rho_{R}\cdot\frac{1}{2}u\left(t\right)^{T}Ru\left(t\right) \tag{3.3}$$

where r is a reference, K and R are coefficient matrices, and ρ_R is an adaptation parameter. The right-hand-side of Eq. 3.3 is a representation of the energy of the states and control inputs at each time instance. The adaptation parameter, ρ_R, balances between system performances and control efforts. The higher the adaptation parameter, the less agile the system movement.

Control input constraints are handled in a variety of ways. A naive clamping method constrains the input signals on the boundaries if the optimal control signals exceed their limits. It is easily applied, without guaranteeing though that the DDP input sequence is an optimal solution. Applying squashing functions on the control inputs is a soft constraint method. Unlike the naive clamping technique, it is expected to produce an optimal solution although, due to the nature of the non-linearity of a squashing function, the DDP approach may not converge when the higher order terms become significant. Tassa, Mansad, and Todorov proposed a quadratic programming approach to address this issue [67].

Unlike control input constraints, state constraints are relatively difficult to realize. In many applications, they are formulated indirectly as a cost function. For instance, if a system state must be within a specific safe range, a running cost function can be formulated as an exponential function for the system state introducing a very high cost below or beyond the designated safety limits.

The optimal control inputs, using DDP, are iteratively calculated at every control step:

$$\delta u^{*}\left(t_{k}\right)=-Q_{uu}^{-1}Q_{u}-Q_{uu}^{-1}Q_{ux}\delta x\left(t_{k}\right) \tag{3.4}$$

The DDP module exhibits a second-order fast convergence rate, as shown in Eq. 3.4.

However, careful treatment is required in the construction of the Hessian matrix, Q_{uu}. Q_{uu} must be invertible; i.e., $Q_{uu} > 0$ is a necessary condition for the application of the DDP algorithm. Moreover, since system dynamics are generally nonlinear, their linear approximation implies that the step size for the update of control inputs in the iterative process must be regularized.

The beauty of DDP is that it provides not only a sequence of optimal control inputs but also an optimal trajectory simultaneously. Since Q is a function of the coefficients K and R, different choices for these coefficients result in different system behaviors, as long as Q_{uu} is invertible and the system is controllable. Therefore, these coefficients are vital to determining the level of trade-offs between system performance and control effort.

An important point to note is that the accuracy of system dynamics is critical for successful control planning. In this regard, the system dynamics should adequately reflect the effect of fault and failure modes. If physical effects are not easily incorporated into the physics-based state-space model or an unknown fault occurs, online system identification is deemed necessary. In this case, a recursive least square regression method for the dynamic system parameters can resolve this issue. Another popular approach is stochastic modeling.

3.1.2 Module 2: Long-term mission capability recovery by a simulation-based prediction model

Long-term mission capabilities are modeled with respect to system states, x, fault states, x_f, and the adaptation parameter, ρ_R, which was introduced in Module 1. Assuming that vulnerable mission capabilities and superior mission capabilities are pre-identified, ρ_R is optimized at every control time step as shown in Eq. 3.5 and Eq. 3.6. Figure 3.4 represents the overall procedure of Module 2.

$$\max_{\rho_R(t)} J\left(x\left(t\right), x_f\left(t\right), \rho\left(t\right)\right) \tag{3.5}$$

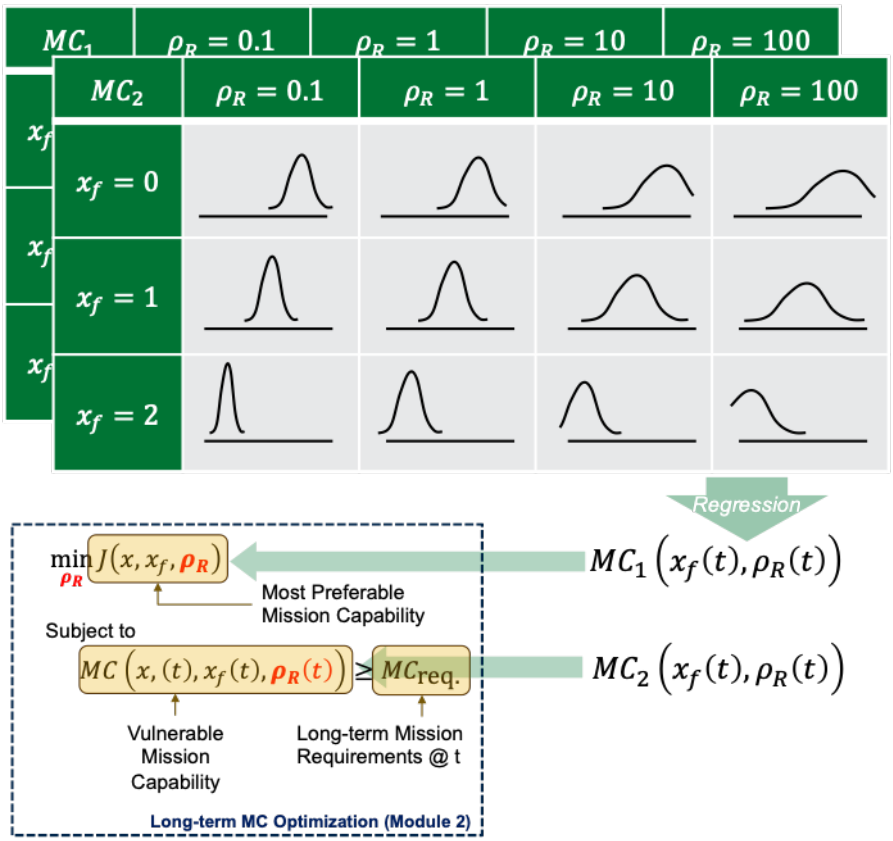

Figure 3.4: Overview of Module 2

subject to:

$$MC\left(x\left(t\right), x_f\left(t\right), \rho_R\left(t\right)\right) \geq MC_{req.} \tag{3.6}$$

where J is an objective function, which is the most superior long-term mission capability, x is a system state, x_f is a fault state, MC is a vulnerable long-term mission capability available, and $MC_{req.}$ is a long-term mission capability required.

J is the most superior long-term mission capability that is impaired and sacrificed due to the fault and the reconfigured control strategy. The objective function, therefore, minimizes the loss of the most superior mission capability after reconfiguration. MC is vulnerable long-term mission capabilities available that need to be extended to satisfy the long-term mission capabilities required, $MC_{req.}$. Here, the mission capability is re-defined as:

- **Definition: Long-term Mission Capability**
  - A measure of the maximum system performance available until the system fails; given the condition of the system during the mission

The long-term mission capability depends on a series of ρ_R in future selection. Assuming that Eq. 3.2 and a mission profile are given and well-known, a system can be simulated by inducing an artificial fault mode. For a simple MC modeling, ρ_R is set to be constant in each simulation episode.

By randomly inducing a fault mode at any mission point, MC at a certain combination of x, x_f, and ρ_R is evaluated in a statistical manner considering (aleatory) uncertainties and noises. For example, by assuming a Gaussian distribution, MC prediction can be parameterized by two statistical parameters, mean (μ_{MC}) and standard deviation (σ_{MC}):

$$MC\left(x, x_f, \rho_R\right) \sim N\left(\mu_{MC}, \sigma_{MC}\right) \tag{3.7}$$

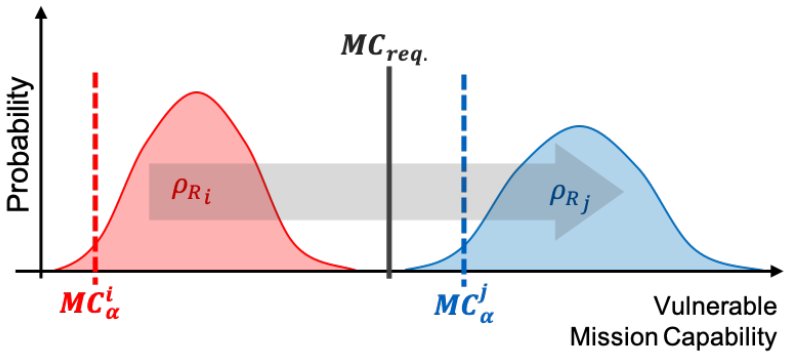

Figure 3.5: Notional representation of MC_α shift by ρ_R

By regression techniques, μ_{MC} and σ_{MC} are regressed with respect to x, x_f, and ρ_R:

$$\begin{aligned} \mu_{MC} &= F_{\mu_{MC}}(x, x_f, \rho_R) \\ \sigma_{MC} &= F_{\sigma_{MC}}(x, x_f, \rho_R) \end{aligned} \tag{3.8}$$

where $F_{\mu_{MC}}$ and $F_{\sigma_{MC}}$ are regression functions.

For a robust adaptation, statistical confidence is introduced instead of μ_{MC}. Therefore, the original constraint equation, Eq. 3.6 can be re-written:

$$MC_\alpha \geq MC_{req.} \tag{3.9}$$

Likewise, the objective function, J, is expressed by a regression function in terms of x, x_f, and ρ_R. These regression mappings described above make the optimization problem simple. Any iteration solver can be applied. A preferable method will be dependent on the system and the mission capabilities. If the objective function is monotonic for ρ_R, then a zero-crossing detection algorithm will find a solution for $MC_\alpha - MC_{req.} = 0$. Figure 3.5 illustrates a notional shift of MC_α by ρ_R.

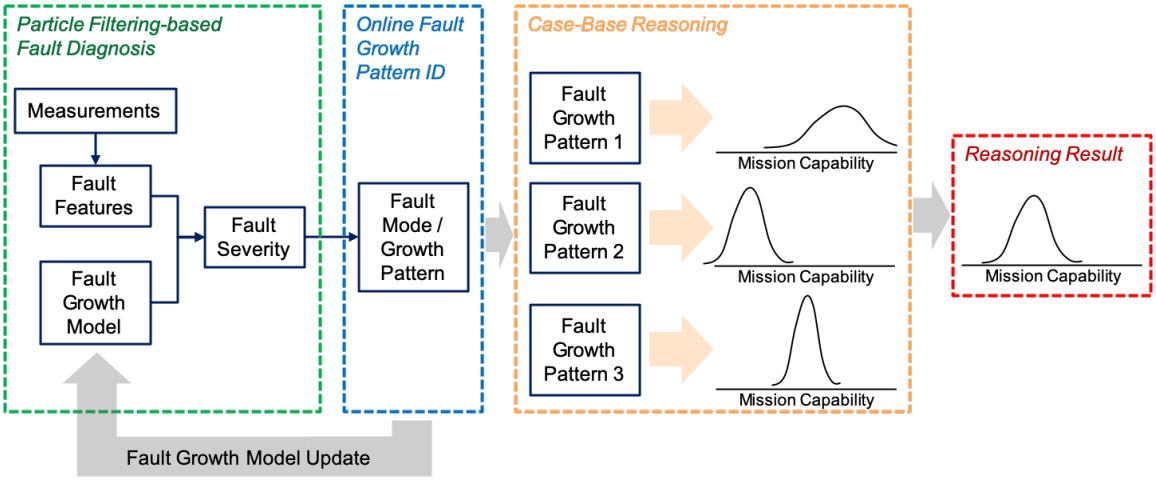

Figure 3.6: Overview of Module 3

3.1.3 Module 3: Situational awareness by particle filtering-based fault diagnosis and CBR

For the better prediction of the mission capability in Module 2, an unknown fault growth rate is identified, and a mission capability prediction model is adjusted accordingly by Case-Based Reasoning (CBR) in Module 3 as depicted in Figure 3.6. A particle filtering-based fault diagnosis approach detects a fault mode and estimates its state. Fault state estimates are used to identify the fault growth rate online. Then, a corresponding mission capability prediction model is reasoned and used in Module 2.

A typical CBR follows a process cycle, called four R's [68]. The four R's are:

1. Retrieve
2. Reuse
3. Revise
4. Retain

Figure 3.7 describes the cycle of CBR process. As a new event arrives, probable solutions stored in the case base are retrieved that are most similar to the new event. If the event perfectly matches one of the cases in the case base - i.e., the incoming problem is

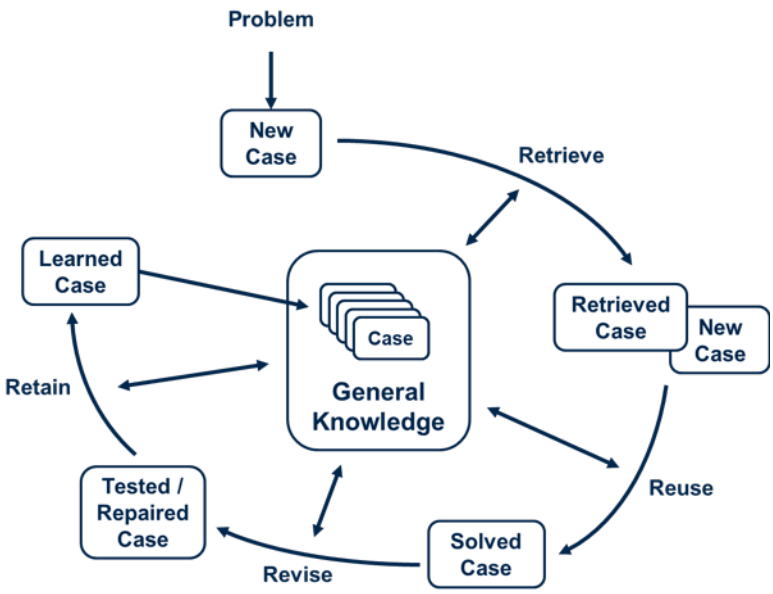

Figure 3.7: A cycle of CBR process (redrawn based on [68])

identified as a case experienced in the past - then, the corresponding solution is applied. If the event does not precisely match any stored cases, a set of predetermined rules, in the form of similarity metrics, are called upon to adapt stored solutions to the new event. The new event forms a new case in the case base.

CBR is composed of four knowledge containers [69, 70].

1. Vocabulary: Data structure and its elements representing the primitive notion
2. Similarity measure: Numerical modeling of similarity
   - Reflexivity: $sim\,(x, x) = 1$
   - Symmetry: $sim\,(x, y) = sim\,(y, x)$
   - Nearest neighbor: m is the nearest neighbor to p if $sim\,(p, m) \geq sim\,(p, m')$
   - Distance function: $d\,(x, y) < d\,(u, v) \leftrightarrow sim\,(x, y) > sim\,(u, v)$
3. Case-base: Experience representation composed of a problem-solution pair
   - $c\,(p, s) \in CB$

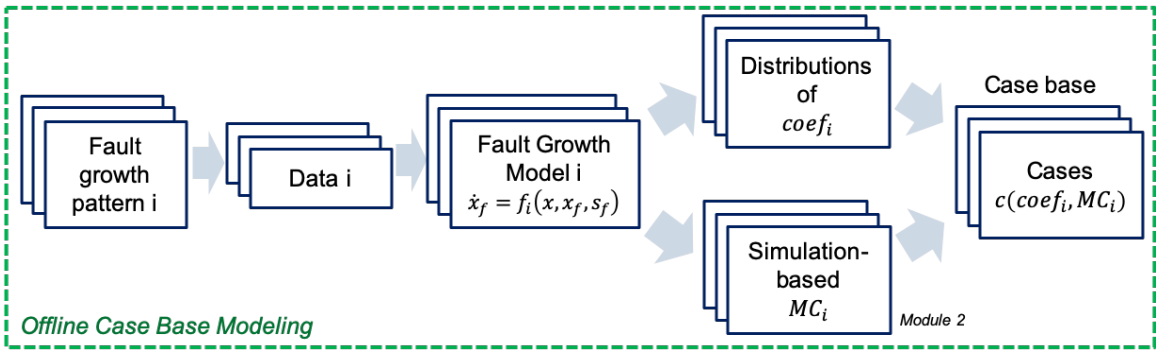

Figure 3.8: Offline case-base modeling procedure

- CB is a case-base, c is a case, p is a problem, and s is a solution

4. Solution transformation: Solution adaptation rules

In Module 3, a fault growth model is translated into a case. By recalling fault dynamics in Eq. 3.2, possible typical fault growth rates are modeled as:

$$\dot{x}_f = F_{f_i}\left(x_f, s_f, \text{coeff.}_i\right) \tag{3.10}$$

where i is a case number and coeff. is a vector of model coefficients.

Assuming that the same fault mode shares the same growth pattern, coef.$_i$ is a vocabulary of a case problem. Physics or test data can determine the structure of the fault growth model. A symbolic regression technique is a candidate, determining the best regression model out of data. Model coefficients - a problem of a case - can also be represented by statistical representation in consideration of estimation uncertainties.

On the other hand, a solution of a case is the mission capability prediction model $MC_i\left(x, x_f, s_f\right)$, which is obtained by simulation-based modeling as described in the previous section. By putting the problem and solution, a case is designed. Figure 5.3 shows a conceptual procedure of offline case-base modeling.

$$c_i\left(\text{coef.}_i, MC_i\left(x, x_f, s_f\right)\right) \in CB \tag{3.11}$$

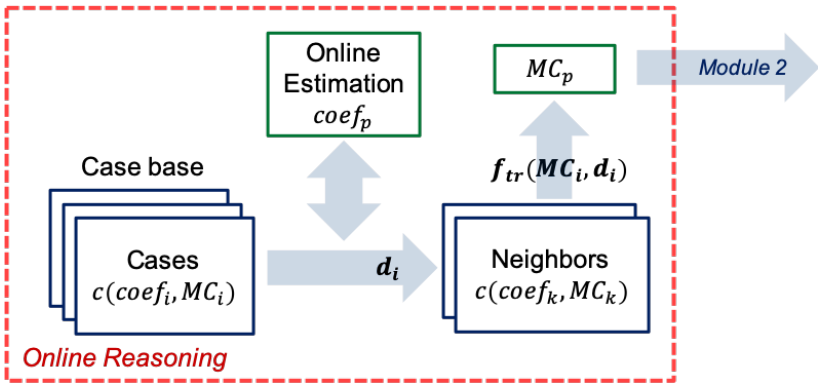

Figure 3.9: Online reasoning procedure

For the similarity measure, the nearest neighbors can be selected by Euclidean distance between coef.$_i$ and coef.$_p$ where coef.$_p$ is a present coef. estimated online. Assuming that a similar case produces a similar result, an inverse distance weighting method can be used for the solution transformation rule. Of course, a selection of a specific method should rely on the characteristics of the application system, fault modes, a fault growth model structure, etc. As an example, the Shepard's interpolation method is shown below [71]:

$$MC_p = \begin{cases} \frac{\sum_{i=1}^{N} w_i(\text{coef.}_p) \cdot MC_i}{\sum_{i=1}^{N} w_i(\text{coef.}_p)}, & \text{if } d\left(\text{coef.}_p, \text{coef.}_i\right) \neq 0, \quad \forall i \\ MC_j, & \text{if } d\left(\text{coef.}_p, \text{coef.}_j\right) = 0 \end{cases} \tag{3.12}$$

where

$$w_i\left(\text{coef.}_p\right) = \left(\frac{1}{d\left(\text{coef.}_p, \text{coef.}_i\right)}\right)^2$$
$$d\left(\text{coef.}_p, \text{coef.}_i\right) = \mid \text{coef.}_p - \text{coef.}_i \mid_2 \tag{3.13}$$

Figure 3.9 depicts an online reasoning procedure.

3.1.4 Assumptions

The proposed technical approach was built upon a set of assumptions. The assumptions provide a guideline for the application of the proposed framework as well as its limitations.

The assumptions can be grouped into four categories, as shown below.

Known

1. Mission and mission requirements are known.
2. The most superior mission capabilities are known.
3. Critical failure modes and corresponding common fault modes for a given system are identified.
4. Vulnerable mission capabilities due to fault modes are known.
5. System dynamics models are known and reasonably accurate.
6. Fault features are known and selected for each fault mode accordingly. $\rightarrow$ Fault features can isolate a fault mode from others.
7. Impact of fault levels on system performances and states are identifiable.
8. Typical fault growth models are identifiable.

Unknown

1. Fault growth models are unknown when a fault mode is detected.

Problems of Interest

1. The proposed framework does not deal with sensor fault modes.
2. Once a fault occurs, it stays unless it is repaired.
3. The severity of a fault grows monotonically.
4. Multiple fault modes do not coincide.
5. Time is not the only vulnerable long-term mission capability domain in the presence of a critical fault mode.

Hardware

1. Proper sensors for health monitoring as well as system controls are on-board.

The *Known* category defines necessary information in order to design the proposed resilience-enhanced reconfigurable control framework. The *Problems of Interest* category defines applications that the proposed framework is applicable. As long as the assumptions are met, the proposed resilience-enhanced reconfigurable control framework will work in any dynamical systems. The *Unknown* is a challenge that the proposed framework needs to address. Finally, the *Hardware* category explains the requirement for sensors on board.

3.2 Research Questions and Hypotheses

This section explains how the proposed technical methods in each module demonstrated in the previous section were chosen out of alternatives addressed by a set of research questions. Hypotheses were formulated accordingly to test the efficacy of the proposed methods.

The proposed resilience-enhanced reconfigurable control framework can be viewed as the advanced version of AFTCS. Therefore, it is good to start with the major components of AFTCS as reviewed in Chapter 2. AFTCS is generally composed of:

1. Fault Detection and Diagnosis (FDD)
2. Reconfiguration Mechanism
3. Reconfigurable controller
4. Command / reference governor

Table 3.2: Proposed resilience-enhanced reconfigurable control functions and requirements

AFTCS functions	Proposed functions	Requirements
FDD	Reasoning	Induction and learning
Command/reference governor and reconfiguration mechanism	Long-term recovery	Performance trade-offs and consideration of consequences
Reconfiguration mechanism and reconfigurable controller	Immediate recovery	Performance trade-offs and learning

The four modules above can be mapped to three fundamental functions. First, FDD is a required capability of the reasoning function since it must be able to identify and be aware of what is going on in the system online. Reasoning, however, should not be limited to fault diagnosis on known fault modes or fault growth patterns. A necessary property of reasoning is to induce conclusions for unknown situations by combining the incoming evidence and a priori knowledge.

Second, the reconfiguration mechanism and command/reference governor modules are mapped to the long-term performance trade-off function in the proposed framework. The proposed framework needs to optimize the operating conditions considering not only the impact of a critical fault mode, but also the changes in mission capability (consequences) due to the modification of a control strategy comprehensively.

Lastly, the reconfigurable controller and mechanism modules are relevant to the immediate recovery function. The immediate recovery function must produce control signals to make the impaired system stable and to help the faulty system follow its references. Just like long-term performance trade-offs, it also needs to consider the loss of performance and its degradation comparing to performance requirements.

The relationships between the proposed resilience-enhanced reconfigurable control functions and AFTCS modules are shown in Table 3.2.

3.2.1 Module 1: Immediate Performance Recovery

Research Question 1-1

What is a proper reconfigurable control method for Module 1?

Module 1 is fundamentally the same as the traditional reconfigurable controller. Therefore, it is a good idea to review popular methods for the reconfigurable controller in order to develop or choose a proper method for the proposed framework. The goal of reconfigurable controllers is to put the degrading operating points back to the original operating point as close as possible. With this regard, reconfigurable control systems typically aim at:

1. Stabilization
2. Equilibrium recovery
3. Output trajectory recovery
4. State trajectory recovery

Zhang and Jiang extensively reviewed literature about reconfigurable fault-tolerant control systems [49]. Based on mathematical design tools, they classified reconfigurable control design methods into controller design approaches, reconfiguration mechanisms, and applied system characteristics for typical AFTCS designs. This classification is beneficial to distinguish characteristics of different reconfiguration methods and choose a proper one based on the required properties of the proposed framework.

In terms of the design approach, online automatic redesign methods are more flexible to unknown fault modes than pre-computed control laws. Moreover, when a fault can grow over time, the intensity of the impact of a fault varies as a system operates; thus, a handful of pre-calculated controllers would not be able to handle it properly.

For reconfigurable mechanism, each approach is briefly reviewed to compare them as followings: First, a switching approach is to choose proper control gains or control laws among multiple models for healthy and faulty plant conditions considered. It is also called projection approach in another literature because this reconfiguration mechanism projects an incoming situation to a known scenario [72]. It can provide expected and satisfactory results for modeled conditions. However, since it requires pre-computed control laws by its nature, a switching approach itself is not enough to cope with degrading system performances.

As a second approach, a model-matching approach defines a reference model, which satisfies nominal control specifications. Let us consider a Linear Time-Invariant (LTI) system:

$$x\left(k+1\right) = Ax\left(k\right) + Bu\left(k\right) \tag{3.14}$$

where $x\left(k\right) \in \mathbb{R}^n$ is a system state vector, $u\left(k\right) \in \mathbb{R}^m$ is a control input vector, and $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ are system property matrices. Suppose that a classical feedback control law is used,

$$u\left(k\right) = -Kx\left(k\right) \tag{3.15}$$

where K is a state gain matrix. If K satisfies nominal control specifications, say K^*, Eq. 3.14 can be rewritten as:

$$x\left(k+1\right) = \left(A - BK^*\right)x\left(k\right) \equiv M^*x\left(k\right) \tag{3.16}$$

M^* is then a reference model. Now, as a fault occurs in a plant or an actuator, A or B changes: A_f or B_f. In turn, a closed-loop model of the faulty system can be no longer the same as the reference model, M^*. Therefore, an adaptive routine finds K^{f^*} making a closed-loop model be M^* for A_f or B_f. In order to get the reconfigured control gain,

matrix inversion is inevitable. Therefore, a Pseudo Inverse (PI) method is popularly used.

However, an exact model-matching approach does not guarantee system stability after reconfiguration; thus, an admissible model-matching approach was proposed by Staroswiecki [73] and extended for a Linear Time-Varying (LTV) system by de Oca et al. [74]

A third approach is a model following. A model-following approach is similar to a model-matching, but a model-following approach makes system states or outputs follow reference states or outputs [75]. It distinguishes a reference model and a plant model. A reference model is:

$$\dot{x}_m = A_m x_m + B_m u_m \tag{3.17}$$

A plant model is

$$\dot{x}_p = A_p x_p + B_p u_p \tag{3.18}$$

Then, it chooses control gains to minimize an error, $e = x_m - x_p$. Like a model-matching approach, PI method is widely used.

Even though both model-matching and model-following methods are well-targeting stabilization and recovery, there are several limitations. First, they are limited to linear systems. Second, since it mainly focuses on reference tracking, it is difficult to embed diagnostic and prognostics knowledge in these methods. Gao and Antsaklis even argued that a model-following approach does not require fault diagnostic routine in their framework [75].

An optimization approach, on the other hand, is to solve an optimization problem to get the best control inputs. The basic idea is to make a trade-off between system performances and control efforts by minimizing (or maximizing) an objective function. A typical form of the cost function is an integral of a linear combination of quadratic terms for states and

control inputs for a certain period as shown in Eq. 3.19.

$$min_{u(t)} \int_0^T \left(x\left(t\right)^T Qx\left(t\right) + u\left(t\right)^T Ru\left(t\right) \right) dt \tag{3.19}$$

where Q and R are positive semi-definite weighting matrices balancing between state errors and control efforts. This approach does not explicitly show the quality of stability of a system, but the cost function penalizes violations of reconfiguration goals (stability, equilibrium, output, and state trajectory recoveries). Therefore, as long as a (faulty) system is controllable, it finds optimal control inputs [72]. The most significant benefit of this approach is its flexibility to embrace diagnostic and prognostic knowledge into the formulation. The main drawback of this approach is the computational requirement.

Lunze et al. also suggested a fault hiding approach for control reconfiguration [72]. What distinguishes a fault hiding approach from the other approaches above is that a fault hiding approach uses a nominal controller as is in the presence of a fault. Instead, it attaches virtual sensors or actuators that accommodate sensor measurements or control input signals to produce a designed control quality. Thus, it does not redesign a controller. However, it does not have trade-off capability, but only signal manipulations.

Table 3.3 summarizes reconfigurable control mechanisms reviewed above and compares them based on requirements for the proposed framework. It is shown that optimal control-based approach fits well to the problem of interest. Of course, it does not mean that only one approach must be chosen; instead, in practice, a combination of them can result in better outcomes than an independent approach [49].

One of the most popular optimization-based reconfiguration methods is a Linear Quadratic (LQ)-based method. It is an automatic way of getting feedback gains. Assuming that Eq. 3.14 is a system model and Eq. 3.15 is a control law stabilizing the system, P is a solution matrix of algebraic Ricatti equation (a continuous-time case) for Eq. 3.19 considering

Table 3.3: Comparisons of reconfigurable control approaches

Approach	Automatic Redesign	Trade-off	Prognosis
Switching	×	○	×
Matching	○	×	×
Following	○	○	×
Optimizing	○	○	○
Fault Hiding	○	×	×

infinite horizon:

$$K = R^{-1}B^{T}P \tag{3.20}$$

$$0 = A^{T}P + PA - PBR^{-1}B^{T}P + Q \tag{3.21}$$

A solution to the Ricatti equation is achieved by matrix factorization or iterative process of the equation. One of the drawbacks of this approach, however, is the fact that there is no explicit relationship between the quality of stability and parameter matrices. It means that it is necessary to decide parameter values to obtain "optimal" control gains, which satisfies stability requirements. Besides, this approach is mostly limited to linear systems only. Therefore, pole placement methods with full state feedback are generally preferred to a LQ-based approach.

Another optimization-based approach is Model Predictive Control (MPC), also known as Receding Horizon Control (RHC). MPC has been a popular method for reconfigurable control in recent researches. Figure 3.10 compared MPC with other popular methods of control reconfiguration, and MPC has more excellent properties over other methods. A key concept of MPC is to obtain an open-loop optimal control input sequence for a finite control horizon by solving Eq. 3.19, but only apply the first (or two) control input(s) to the system, measure the following measurements, and repeat to solve the optimal control

Method	Failures		Robust	Adaptive	Fault Model			Model Type	
	Actuator	Structural			FDI	Assumed	Constraints	Linear	Nonlinear
Multiple Model Switching and Tuning (MMST)		•		•	•			•	
Interacting Multiple Model (IMM)		•		•	•		o	•	
Propulsion Controlled Aircraft (PCA)	•		o			•		•	•
Control Allocation (CA)	•					•	o	•	
Feedback linearization	•	•		•	•				•
Sliding Mode Control (SMC)	o[1]	•	•[2]				•		•
Eigenstructure Assignment (EA)		•				•		•	
Pseudo Inverse Method (PIM)		•				•		•	
Model Reference Adaptive Control (MRAC)		•		•	•			•	o
Model Predictive Control (MPC)	•	•	o	o	•	•	•	•	•

Figure 3.10: Reconfigurable control method comparison [79]

at every discrete control time step. Because of this, MPC implicitly produces closed-loop control characteristics.

One of merits of MPC is the fact that it can solve for a constrained system, and the system of interest does not have to be linear. It enables diagnostics and prognostics knowledge to be successfully considered in the objective function or constraints. Also, MPC has been proven to stabilize a system asymptotically in general [76, 77]. The proofs showed that the total cost function, Eq. 3.19, is a Lyapunov function of a system. Appendix B presents the proof from [78].

In order to solve MPC optimal control, Dynamic Programming (DP) approach can be used at each control time for the finite receding horizon if the system dynamics is relatively well-known. DP requires the environment having Markov property. Markov property is observed if a certain state is solely dependent on the previous state and action. Its formal definition is:

$$p\left(s', r\right) = Prob\left(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a\right) \tag{3.22}$$

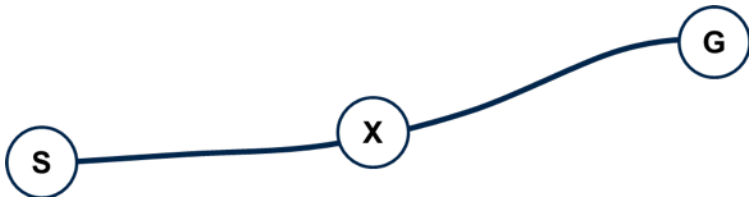

Figure 3.11: Principle of Optimality. If $\tilde{SG}$ is optimal, then $\tilde{XG}$ is optimal at any point, X, on the optimal path, $\tilde{SG}$

for all s', s, a, and r. s and s' are two different states, a is an action, and r is a reward at a certain state, s,

Most state-space model follows Markov property. A general discrete dynamics model is expressed as:

$$x\left(t_{k+1}\right) = f\left(x\left(t_k\right), u\left(t_k\right)\right) + \omega\left(t_k\right) \tag{3.23}$$

where x is a state vector, u is a control input vector, $f\left(\cdot\right)$ is a transition function, and is process noise. For instance, velocity and position (states) of aircraft at a certain point of time only rely on velocity and position and a control input at the previous time step. In this case, if a state is defined only by vehicle positions, then it no longer possesses the Markov property because a history of positions is necessary to identify the next position. Then the running environment with the Markov property is called the Markov Decision Process (MDP).

Assuming the MDP, DP utilizes the Principle of Optimality or also known as Bellmans Principle [80]. Richard Bellman stated the Principle of Optimality as:

> "An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision." Bellman (1957), page 83.

The basic idea of the Principle of Optimality is depicted in Figure 3.11. If the path, $\tilde{SG}$, is an optimal path, then any path, $\tilde{XG}$ is optimal where the point X is on the path,

$\tilde{SG}$. Based on the Principle of Optimality, a large class problem can be solved at once by solving a small problem within the larger class [80]. Suppose that a general nonlinear cost function, $V(\cdot)$, within a fixed finite time in the discrete time domain is expressed by:

$$V\left(x\left(t_0\right), t_0\right) = \int_{t_0}^{t_f} l\left(x\left(\tau\right), u\left(\tau\right), \tau\right) d\tau + \Phi\left(x\left(t_f\right), t_f\right) \tag{3.24}$$

where t_0 is a start time, t_f is a final time, l is a running cost (or an immediate cost), and Φ is a final cost. Solving the optimal control problem is to find the sequence of control inputs, $[u\left(t_0\right), u\left(t_1\right), \cdots, u\left(t_f\right)]$, which minimizes $V\left(x\left(t_0\right), t_0\right)$. This large problem, which considers the entire mission time, can be reformulated into a set of small recursive problems as follows:

$$V\left(x\left(t_k\right), t_k\right) = \min_{u(t_k)} \left[\int_{t_k}^{t_k+\delta t} l\left(x\left(t_k\right), u\left(t_k\right), t_k\right) dt + V\left(x\left(t_k + \delta t\right), t_k + \delta t\right) \right] \tag{3.25}$$

Equation 3.25 means that a cost at any time, $V\left(x\left(t_k\right), t_k\right)$, can be defined as a sum of the immediate cost at t_k and the cost at next time step, $V\left(x\left(t_k + \delta t\right), t_k + \delta t\right)$. The boundary condition of this problem is at the final time, t_f:

$$V\left(x\left(t_f\right), t_f\right) = \Phi\left(x\left(t_k\right), t_f\right) \tag{3.26}$$

Therefore, the optimal control problem is solved from the final time to the initial time backward.

DP assumes that an accurate state transition model exists. (The model is also called a plant or environment model.) Suppose that state-space system dynamics is given as:

$$\frac{dx}{dt} = F\left(x\left(t\right), u\left(t\right)\right) \tag{3.27}$$

where $u \in R^n$ is a control input (desired force/torque) vector, and $x \in R^m$ is a state vector.

If we manipulate Eq. 3.25,

$$V\left(x\left(t_k\right),t_k\right)=\min_{u(t_k)}\left[\int_{t_k}^{t_k+\delta t} l\left(x\left(t_k\right),u\left(t_k\right),t_k\right)dt+dV+V\left(x\left(t_k\right),t_k\right)\right] \tag{3.28}$$

where

$$dV=V\left(x\left(t_k+\delta t\right),t_k+\delta t\right)-V\left(x\left(t_k\right),t_k\right)=\frac{\partial V}{\partial t}dt+\frac{\delta V}{\delta x}dx \tag{3.29}$$

Then,

$$0=min_{u(t_k)}\left[\int_{t_k}^{t_k+\delta t} l\left(x\left(t_k\right),u\left(t_k\right),t_k\right)dt+\frac{\partial V}{\partial t}dt+V_x^T\cdot F\left(x\left(t_k\right),u\left(t_k\right)\right)dt\right] \tag{3.30}$$

where $V_x^T=\frac{\delta V}{\delta x}$. Finally,

$$-\frac{\partial V}{\delta t}=min_{u(t_k)}\left[l\left(x\left(t_k\right),u\left(t_k\right),t_k\right)+V_x^T\cdot F\left(x\left(t_k\right),u\left(t_k\right)\right)\right] \tag{3.31}$$

Equation 3.31 is a partial differential equation, also known as Hamilton-Jacobi-Bellman (HJB) equation. Finding an optimal control policy means solving HJB equation. The Riccati equation by the LQ-based optimal control approach is a special case when a cost function is formulated by a linear-quadratic form with full knowledge of linear system dynamics. However, solving a general PDE HJB equation is difficult or sometimes impossible to get a solution analytically [Lewis]. Thus, DP is typically solved by iterative processes: value function iteration, policy iteration, and policy search methods. System models are used to query data points for the iterative processes. By the nature of DP, DP is typically viewed as an offline learning approach dynamical system controls. Therefore, it is not easy to apply DP for online and real-time controls when system dynamics vary over time (due to fault effects and recovery strategies).

To make DP applicable to practical problems, Differential Dynamic Programming (DDP) solves a local trajectory optimization problem based on Bellmans dynamic programming principle. By the nature of the Bellmans principle, the cost values are propagated backward in time if we know $V\left(x\left(t_f\right), t_f\right)$. The Bellmans principle in discrete-time domain is expressed in Eq. 3.32:

$$V\left(x\left(t_k\right), t_k\right) = \min_{u(t_k)} \left[l\left(x\left(t_k\right), u\left(t_k\right), t_k\right) \Delta t + dV + V\left(x\left(t_{k+1}\right), t_{k+1}\right)\right] \tag{3.32}$$

Suppose that $Q\left(x\left(t_k\right), u\left(t_k\right), t_k\right) = l\left(x\left(t_k\right), u\left(t_k\right), t_k\right) \Delta t$. Sufficient conditions of optimal solutions at $u^*\left(t_k\right)$ are $\frac{\partial Q(u^*)}{\partial u} = 0$ and $\frac{\partial^2 Q(u^*)}{\partial u^2} > 0$. Since DDP solves a problem with the second derivatives of Q, cost functions must be second order differentiable.

By the quadratic approximation of $Q\left(x\left(t_k\right), u\left(t_k\right), t_k\right)$ and the linear approximation of the system dynamics, an optimal control input correction, $\delta u^*\left(t_k\right)$, can be expressed as in Eq. 3.33. Detail derivations of the solution were derived, and typical canonical examples were solved in Appendix C.

$$\delta u^*\left(t_k\right) = -Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} \delta x\left(t_k\right) \tag{3.33}$$

where $\delta\cdot$ is a small variation and subscripts mean partial derivatives. Matrices, Q and its derivatives, are functions of V, l, and their derivatives. They are calculated by backward propagation from t_f to t_0 along with the initial u and x. Then, $\delta u^*\left(t_k\right)$ can be obtained by the forward-sweep process, updating u and x. These backward and forward sweeps iterate until convergence.

Figure 3.12 depicts how DDP works to find optimal control for a trajectory following task. It starts from a random initial path coming from a random sequence of control inputs, and gradually converges to a reference trajectory that a system has to follow.

Based on the review of alternative methods for Module 1 of the proposed resilience-

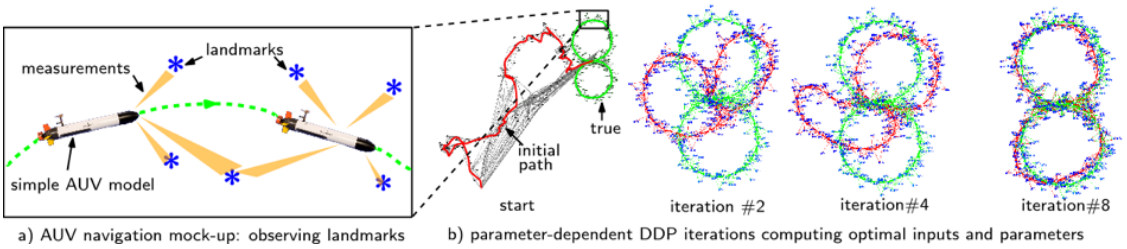

Figure 3.12: An successful example of DDP and its convergence [81]

enhanced reconfigurable control framework, MPC-based AFTCS solving optimal control problems by the DDP method seems to be a reasonable choice.

Research Question 1

What is a proper reconfigurable control method for Module 1?

Possible Answer

An MPC-DDP method is suitable for Module 1.

One of the most significant challenges using DDP in MPC was the computational burden because it should solve the optimization problem at each control step. Thus, in the past, MPC was only applicable to slow dynamics. However, improvement of computational power has alleviated this issue, and so MPC-DDP framework has been successfully utilized in UAS controls [67]. Another challenge in MPC-DDP is model accuracy because DDP is a model-based control approach. If the system dynamics model is not accurate enough, solutions from DDP may not be able to produce optimal control inputs. Therefore, in this study, it is assumed that system properties and dynamics are relatively well-known and accurate.

Now, one thing that has not been shown is the applicability of MPC-DDP, recovering trajectories in the presence of a critical fault mode, as an AFTCS. It led to Hypothesis 1 below:

Hypothesis 1

If MPC-DDP is used for Module 1, then the system trajectory will be recovered in the presence of a critical fault mode.

3.2.2 Module 2: Long-term mission capability trade-off

Research Question 2-1

How to obtain an optimal solutions from Module 1 considering long-term mission capability?

One way of adjusting system performances through MPC-DDP is to modify references online directly. It is one of the ways that past researches proposed, according to the literature review in Chapter 2. Drozeski used linear programming for the reconfigurable path re-planning. The re-planning in his approach, however, was not aiming at long-term mission capability [60]. Tang et al. applied D^* algorithm in a ground vehicle application to optimize a path online considering different optimality criteria: mission time, RUL, and easiness of travel [64]. It addressed the impact of the reconfigured path on mission capability but did not combine it with control reconfiguration.

As discussed in the previous section, DDP generates optimal paths based on the objective function as well as the optimal control sequence. Let us revisit an objective function

of DDP.

$$V\left(x\left(t_0\right),t_0\right)=min_u\left[\int_{t_0}^{t_0+T} l\left(x\left(\tau\right),u\left(\tau\right),\tau\right)d\tau+\Phi\left(x\left(t_0+T\right),t_0+T\right)\right] \tag{3.1 revisited}$$

The most primitive way of long-term performance optimization is to introduce a large finite time window, T. DDP solves the optimization problem for the period of T. If T covers the entire mission time, DDP will be able to find the optimal solutions covering the entire mission.

However, choosing a long time window is practically not a good option. As pointed out in the previous section, the MPC-DDP method can be already computationally burdensome to solve in real time because of an iterative solution approach. A long time window will add more computational burden to it. Moreover, choosing the right T is another problem. Usually, T is set to be constant in an MPC-DDP formulation. Based on an operating point and system health conditions, mission time can change accordingly. In order to choose the right T, T needs to be a variable of the optimization problem, but it will cause another complication. It makes a long time window out of an option.

Another option would be to adjust weights in the cost function. A typical form of the running cost, l, is:

$$l\left(x\left(t_k\right),u\left(t_k\right),t_k\right)=\frac{1}{2}\left(x\left(t_k\right)-r\left(t_k\right)\right)^T K\left(x\left(t_k\right)-r\left(t_k\right)\right)+\frac{1}{2}\left(u\left(t_k\right)^T Ru\left(t_k\right)\right) \tag{3.34}$$

Typically, the coefficient matrices, K and R, are design choices by the desirability of control performances. The first term on the right-hand side of the running cost function is a soft constraint for states, x. K implicitly governs the level of acceptability of performance errors. The higher, the stricter. The second quadratic term is a regulation term for control

effort. The higher R, the slower control responses. Changing the cost coefficients online can reconfigure control strategy so that long-term mission capability can be managed.

This approach, however, may lead to a case where DDP cannot find its solution. As identified in the previous section, Q_{uu} is required to be semi-positive definite at all times. According to Appendix C, Q_{uu} is a function of K and R. A particular combination of cost coefficients can cause Q_{uu} to be negative definite during the online adaptation, and then optimal control solutions cannot be found.

Instead of manipulating the coefficient matrices directly, an additional parameter, ρ_R can be introduced in the running cost function as shown in Eq. 3.3.

$$l\left(x\left(t\right),u\left(t\right),t\right)=\frac{1}{2}\left(x\left(t\right)-r\left(t\right)\right)^{T}K\left(x\left(t\right)-r\left(t\right)\right)+\rho_{R}\cdot\frac{1}{2}u\left(t\right)^{T}Ru\left(t\right) \tag{3.3 revisited}$$

What ρ_R does is to adjust the level of penalty, which is imposed on control efforts. It determines which cost term, the state error, or the control effort, contributes more to the total running cost. Presumably, the high ρ_R will allow state deviations during the operation by suppressing the use of the faulty component. Unlike manipulating the entire cost coefficient matrices, the range of ρ_R can be easily determined so that Q_{uu} is maintained to be invertible.

Reviews of the alternatives above suggest that introducing an adaptation parameter, ρ_R, in the running cost equation is a preferable option. MPC-DDP generates an optimal path, so optimizing each state reference can be somewhat redundant. Directly manipulating T, K, or R can be impractical. Adjusting ρ_R is easier and presumably enables mission capability trade-offs. However, it still needs to be proven that the adaptation parameter affects long-term mission capabilities with Hypothesis 2.

Research Question 2-1

How to obtain optimal solutions from Module 1 considering long-term mission capability?

Possible Answer

By introducing an adaptation parameter, ρ_R, in a running cost of the DDP formulation

Hypothesis 2

If the adaptation parameter varies, then long-term mission capabilities will be adjusted.

Now, the question is how to determine optimal ρ_R when a fault happens.

Research Question 2-2

How to determine the adaptation parameter considering long-term mission capability online?

The primary purpose of Module 2 is to maintain mission capabilities in the presence of a critical fault mode until the faulty system completes its mission goals. The mission is said to be satisfied if the system can produce the mission capabilities exceeding mission

requirements during mission time as shown Eq. 3.35.

$$\text{Mission requirements} < \text{Mission capabilities } \forall t \in [t_0, t_F] \tag{3.35}$$

where t_0 is an initial time, and t_F is a mission terminal time. It means that the proposed resilience-enhanced reconfigurable control framework needs to find a control strategy that enables vulnerable mission capabilities to be extended beyond mission requirements. However, it inevitably comes with the expense of performance sacrifices.

With this regard, the long-term trade-off can be viewed as another optimal control routine. The objective function should represent given mission goals as a function of a set of states and actions. What distinguishes this module from a common optimal control is that this routine more focuses on consequences rather than system stability. (System stability should be guaranteed by Module 1.) In order to improve the consequence-based resilience metrics, consequence-related states should be considered in the objective function and compared to the requirements. For example, a system RUL is a typical consequence-related performance metric, and it needs to be compared to mission time remained when RUL is defined in the time domain. Control actions, on the other hand, will be the adaptation parameter.

Reinforcement Learning

Reinforcement Learning (RL) is a recently highlighted optimal control method. RL assumes that state transition dynamics is unknown, or it is challenging to model because of uncertainty. (In RL-related researches, state transition dynamics is called an environment model more often.) For instance, in a game environment, one player cannot accurately predict the behaviors of the opposite player. The opposite player is viewed as a part of the environment, and so it is considered as a source of uncertainty.

In such an unknown environment, an RL-based control method enables an agent to learn optimal policy by interacting with the environment [82]. It finds a way that a cumulative

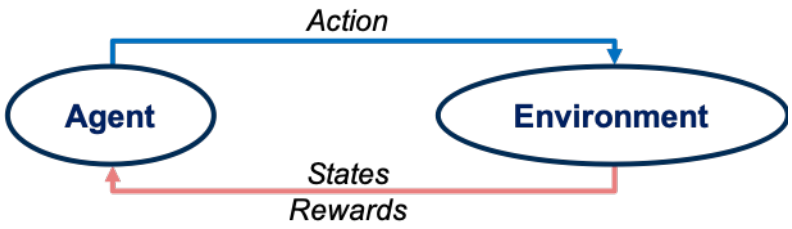

Figure 3.13: A concept of reinforcement learning

reward over a given mission to be maximized. Based on the concept of trial-and-error, experiences are formed into a value or a control policy function as depicted in Figure 3.13.

For episodic cases, one can operate the system with a certain control policy, and then evaluate its value from total discounted rewards. If the environment is uncertain, its value incrementally converges as episodes are repeated. It is called the Monte Carlo (MC) learning method. A total discounted reward at a certain time, t, is defined as [82]:

$$R_t = \sum_{i=t}^{T-1} \gamma^{i-t} \cdot r_{i+1} = r_{t+\gamma} \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \cdots + \gamma^{T-t-1} \cdot r_T \tag{3.36}$$

where R_t is a total discounted reward at t, r_t is an immediate reward at t, T is a final time, and $\gamma \in [0, 1]$ is a discount factor. An update rule of the value function by MC is:

$$V(s_t) \leftarrow V(s_t) + \alpha (R_t - V(s_t)) \tag{3.37}$$

where $V(s_t)$ is a state value at t, and α is a learning rate (also called a step-size parameter).

MC learning algorithm initializes $V(s_t)$ and incrementally updates it by the difference between the collected R_t from an episode and $V(s_t)$. We call R_t a target.

In a non-episodic environment, on the other hand, the value function is estimated at each control action performed on the fly. It is called Temporal Difference (TD) learning method. By evaluating a difference between estimated and actual rewards (costs), the state

value is updated as shown below:

$$V(s_t) \leftarrow V(s_t) + \alpha\left(r_{t+1} + \gamma \cdot V(s_{t+1}) - V(s_t)\right) \tag{3.38}$$

This formulation involves one-step estimation, $r_{t+1} + \gamma \cdot V(s_{t+1})$, for future rewards and values. It is also called a backup. The term, $r_{t+1} + \gamma \cdot V(s_{t+1}) - V(s_t)$, is called TD-error or temporal difference. It depends on the state and reward at $t+1$; thus, the update of $V(s_t)$ can be done at a time, $t+1$. Therefore, it does not have to be episodic training. Of course, TD can be used in the episodic system environment as well, but the value updates do not have to wait until each episode is finished.

In terms of the concept of updating rules, MC and TD learnings use a different target from each other; MC uses R_t, and TD uses $r_{t+1} + \gamma \cdot V(s_{t+1})$. R_t is an unbiased estimator of a true value function, but it has high variance. Thus, it takes a relatively long time to converge. TD uses $r_{t+1} + \gamma \cdot V(s_{t+1})$ as a biased estimator, but it has low variance. Theoretically, TD results can be biased even though TD converges faster than MC. In order to alleviate these limitations, the concept of eligibility trace was introduced. The basic idea is to update every state, which has been experienced during learning, with a decaying parameter, $\lambda \in [0, 1]$. The more states are recently experienced, the greater corrections are applied in the value update. It is called TD() learning. The update rule is:

$$V(s_t) \leftarrow V(s_t) + \alpha \cdot \delta_t \cdot e_t \tag{3.39}$$

where

$$\begin{aligned} \delta_t &= r_{t+1} + \gamma \cdot V(s_{t+1}) - V(s_t) \\ e_t(s) &= \begin{cases} \gamma \cdot \lambda \cdot e_{t-1}(s) + 1, & \text{if } s = s_t \\ \gamma \cdot \lambda \cdot e_{t-1}(s), & \text{otherwise} \end{cases} \end{aligned} \tag{3.40}$$

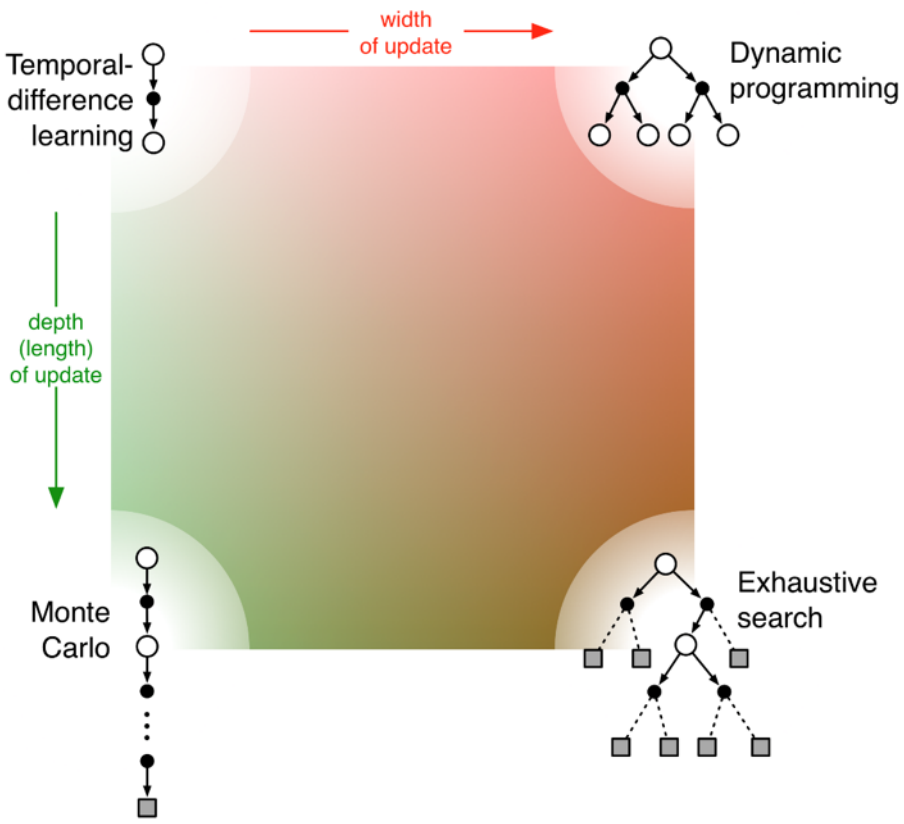

Figure 3.14: A summary and comparisons of DP, TD, MC and exhaustive search methods [82]

$e_t(s)$ is an eligibility trace. It is a function of all states and decays over time. For both extremes, TD(0) is the simplest TD method, and TD(1) is similar to the MC method. However, it does not necessarily mean that TD(1) is only applicable to an episodic environment.

Figure 3.14 depicts a pictorial summary and comparisons of DP, TD, MC, and Exhaustive search methods. (Exhaustive search is not considered as DP or RL, but it explains the relationships of DP, TD, and MC.) Open circles are states, solid circles are actions, and grey squares are terminals. DP considers every state and action available at the next time step. TD only has a single backup. MC samples up to a terminal (a single and long path) and updates values for all experienced states and actions. Exhaustive search samples every possible state and action. TD(λ) falls between TD and MC.

MC, TD, and TD(λ) help estimate a value function. The real problem here is to choose

optimal control policies maximizing the corresponding value. In many cases, an action value function, $Q(s_t, a_t)$ is introduced to solve the optimal control problem instead of $V(s_t)$.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\text{“Target”} - Q(s_t, a_t)) \tag{3.41}$$

where $a \in A$ is a control action within a control action set A.

In order to find optimal controls, there exist briefly two approaches, and each of them can be categorized into three different approaches [83, 84]:

1. Value-based approach
    (a) Value iteration method
    (b) Policy iteration method
2. Policy-based approach
    (a) Gradient-based method
    (b) Non-gradient-based method
3. Actor-critic approach

The value-based approach improves $Q(s_t, a_t)$, and then the optimal control policy is implicitly determined. The value iteration method iteratively finds the optimal value function starting from an arbitrary action-value function, and then derives the optimal control policy from it. The policy iteration method, on the other hand, repeats the iterative process of policy evaluation and policy improvement: starting from an arbitrary control policy, evaluating it by an action value, and improving it until the action value (or control policy) converges. Typically, the value-based approach is suitable for a discrete environment with discrete states and control inputs. In a continuous domain, Value Function Approximation (VFA) method parameterizes a value function with a feature vector, $\phi(s, a)$, and a

parameter vector, w. MC, TD, or TD(λ) can be directly applicable to determine the parameter vector, w. However, making $\phi(s, a)$ is not straightforward, and certain value function approximators can cause convergence issues [84].

The policy-based approach, on the other hand, directly parameterizes the control policy, and finds the optimal policy by optimization processes for an objective function, $J(\theta)$, for the parameter, θ. Typically, the control policy can be parameterized as a stochastic control policy:

$$\pi_\theta(s, a) = Pr(a \mid s, \theta) \tag{3.42}$$

where π_θ is a parameterized control policy, which is a probability of actions at states and the parameter θ. If the objective function is not differentiable for θ, then non-gradient methods should be the right choice. Also, non-gradient methods can theoretically find the global optimum. Gradient-based methods tend to find local optima in general. Using the gradient is, however, more efficient than non-gradient methods.

The objective function for the policy-based approach can be formulated as either one of the following:

1. A starting value for an episodic environment:

$$J(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[R_1] \tag{3.43}$$

2. An average value for a continuing environment:

$$J(\theta) = \sum_s d^{\pi_\theta} V^{\pi_\theta}(s) \tag{3.44}$$

3. An average reward per time step:

$$J(\theta) = \sum_s d^{\pi_\theta} \sum_a \pi_\theta(s, a) R_s^a \tag{3.45}$$

where d^{π_θ} is a stationary distribution of π_θ.

Since the total discounted rewards are unknown, they must be estimated by data. MC policy gradient method, also known as REINFORCE, uses episodic data, and iteratively improves the control policy by using the gradient, $\nabla_\theta J(\theta)$ [85]. The learning process is performed when each episode ends. By using MC policy gradient method, the policy parameter θ is updated at each episode by:

$$\Delta\theta_t = \alpha \cdot \nabla_\theta \ln \pi_\theta left(s_t, a_t right) \cdot V_t \tag{3.46}$$

The important issue of MC policy gradient method is the speed of convergence due to the nature of the MC process: high variance in R_1. Also, it is suitable for the episodic environment. In order to overcome the issue in MC policy gradient method, an actor-critic approach aims at reducing the variance of the value estimates. The actor-critic approach also parameterizes the action-value function and uses this estimate instead of V_t. When $Q_w(s,a)$ is directly estimated, we call it Q Actor-Critic method.

$$\Delta\theta_t = \alpha \cdot \nabla_\theta \ln \pi_\theta(s_t, a_t) \cdot Q_w(s_t, a_t) \tag{3.47}$$

In this formulation, the Critic evaluates the action value function by the parameter w, and Actor improves the control policy along the gradient direction while considering $Q_w(s,a)$ at the same time. Major advantages of the actor-critic approach over the policy gradient method are :

1. Faster convergence

2. Applicability to the continuous state and action domain

As derivatives of Q Actor-Critic method, Advantage function, $A(s,a)$, can be used instead of $Q_w(s,a)$ in the Actor. The idea is to introduce the Base function, $B(s,a)$, and update the policy parameter to make the action value greater than $B(s,a)$; i.e., $A(s,a) =$

Table 3.4: RL value function estimation methods

Approach	Episodic?	Target Estimate	Convergence	Memory
MC	Yes	Unbiased, high variance	Slow	Large
TD	No	Biased, low variance	Relatively fast	Small
TD(λ)	No	In between MC and TD	Faster than TD	Moderate

$Q(s,a) - B(s,a)$. Typically, state value function can be chosen as Base function, and then Advantage function can be parameterized as a function of states, not actions. This formulation specifically derives TD and TD(λ) Actor-Critic methods. The details of various Actor-Critic methods are described in (ref).

So far, the derivations of Policy Gradient and Actor-Critic methods were based on the stochastic policy. Limitation of the stochastic policy gradient method is the fact that as the control policy converges to the deterministic policy, the variance of the policy goes to infinity specifically when the action space is continuous [86]. Also, the stochastic policy does not handle actions in continuous space. Silver et al. proposed deterministic policy gradient algorithms as a special case of stochastic policy gradient method.

This subsection has covered a variety of alternatives in RL. Their characteristics are summarized in Table 3.4 and Table 3.5.

According to the observations above, the Actor-Critic method seems to be appealing because in general Module 2 is supposed to deal with states and actions in continuous domains. However, it is still indecisive to decide on a specific method for Module 2. Using value function approximators (parameterized representation) does not guarantee convergence, specifically when non-linear function approximator is used [84]. Also, data efficiency and difficulty in reward design are challenging [87].

Table 3.5: RL control policy learning methods

Approach	Policy	Value Function	Speed	Popular Methods
Value-based	Implicit	Parameterized	Depending on dimensionality	Q-learning, SARSA learning
Policy-based	Parameterized	Sampled	Slow (MC-based)	REINFOCE
Actor-Critic	Parameterized	Parameterized	Fast	Q Actor-Critic, TD Actor-Critic, Natural Actor-Critic

Simulation-based mission capability modeling

As pointed out in [84], RL with nonlinear value function approximation has a chance not to be able to learn an optimal control solution. Even if RL can find an optimal control solution, often, other methods out-perform RL due to the data inefficiency of RL [87]. In this section, an alternative method for Module 2 is proposed.

Determining the adaptation parameter is an optimization process since it requires mission capability trade-off. A notional optimization problem of the long-term mission capability trade-off was formulated as shown in Eq. 3.5 and Eq. 3.6. Equation 3.5 and 3.6 are revisited below:

$$\max_{\rho_R(t)} J\left(x\left(t\right), x_f\left(t\right), \rho_R\left(t\right)\right) \tag{3.5 revisited}$$

subject to:

$$MC\left(x\left(t\right), x_f\left(t\right), \rho_R\left(t\right)\right) \geq MC_{req.} \tag{3.6 revisited}$$

J is the most superior long-term mission capability that is impaired and sacrificed due to

the fault and the reconfigured control strategy. The objective function, therefore, minimizes the loss of the most superior mission capability after reconfiguration. MC is vulnerable long-term mission capabilities available that need to be extended to satisfy the mission capabilities required, $MC_{req.}$. MC is shifted by fault states and the adaptation parameter. $MC_{req.}$ is assumed to be constant. For instance, as claimed in Research Gap 3, as RUL (MC) is extended by ρ_R, the mission time required would also increase. It makes the long-term mission capability optimization problem challenging to solve by failure prognosis only, which predicts RUL in the time domain in general.

This challenge addresses the necessity of comprehensive MC and $MC_{req.}$ mapping functions from x, x_f, and ρ_R. Assuming that Eq. 3.2 and a mission profile are well-known and given, a system can be simulated by inducing an artificial fault mode at any point of the mission. Even though a simulation can never be the same as the real application, the simulation-based approach has its benefits in identifying mission capabilities in the presence of a critical fault in the system:

- **Simulation Benefits**

    1. A fault can be induced at any point in time during the mission.
    2. The fault intensity is controllable.
    3. A failure can be introduced in the simulation. In real applications, safety and cost-related concerns are inevitable.
    4. Simulations are way faster than hardware testing.

As a fault mode is injected in the middle of the mission, the long-term mission capabilities available needs to be evaluated for ρ_R. The long-term mission capability is dependent on not only the choice of ρ_R at t, but also the series of ρ_R until the end of the mission. In a discrete form, the prediction of MC available is represented by Eq. 3.48. According to the definition of the long-term mission capability, the long-term mission capability is evaluated

by simulating the system until the system fails.

$$MC\left(x\left(t_k\right), x_f\left(t_k\right), \overrightarrow{\rho_R}\right) = f_{MC}\left(\overline{x}\left(t_{k+N}\right), t_{k+N}, x\left(t_k\right), t_k\right) \tag{3.48}$$

where t_k is the current time step, t_{k+N} is a system failure time after N steps, $\overrightarrow{\rho_R} = [\rho_R\left(t_k\right), \rho_R\left(t_{k+1}\right), \cdots, \rho_R\left(t_{k+N-1}\right)]$, and $\overline{x}\left(t_{k+N}\right)$ is a predicted state at t_{k+N}. f_{MC} is a mapping from system states to mission capability. For instance, the predicted mission time is $t_{k+N} - t_k$, and the predicted distance available is Distance Traveled $\left(t_{k+N}\right)$ − Distance Traveled $\left(t_k\right)$.

Estimated system states at t_{k+N}, $\overline{x}\left(t_{k+N}\right)$, is derived by the state propagation through simulations:

$$\begin{aligned}
\overline{x}\left(t_{k+N}\right) &= F\left(\overline{x}\left(t_{k+N-1}\right), \overline{x}_f\left(t_{k+N-1}\right), \overline{u}\left(t_{k+N-1}\right), t_{k+N-1}\right) \\
\overline{x}_f\left(t_{k+N}\right) &= F_f\left(\overline{x}_f\left(t_{k+N-1}\right), \overline{s}_f\left(t_{k+N-1}\right), t_{k+N-1}\right) \\
\overline{s}_f\left(t_{k+N-1}\right) &= F_s\left(\overline{x}\left(t_{k+N-1}\right), \overline{u}\left(t_{k+N-1}\right), t_{k+N-1}\right) \\
\overline{u}\left(t_{k+N-1}\right) &= F_{MPC}\left(\overline{x}\left(t_{k+N-1}\right), \overline{x}_f\left(t_{k+N-1}\right), \rho_R\left(t_{k+N-1}\right)\right) \\
&\vdots \\
\overline{x}\left(t_{k+1}\right) &= F\left(x\left(t_k\right), x_f\left(t_k\right), u\left(t_k\right), t_k\right) \\
\overline{x}_f\left(t_{k+1}\right) &= F_f\left(x_f\left(t_k\right), s_f\left(t_k\right), t_k\right) \\
s_f\left(t_k\right) &= F_s\left(x\left(t_k\right), u\left(t_k\right), t_k\right) \\
u\left(t_k\right) &= F_{MPC}\left(x\left(t_k\right), x_f\left(t_k\right), \rho_R\left(t_k\right)\right)
\end{aligned} \tag{3.49}$$

where $\bar{\cdot}$ is a predicted estimate, x is a system state, x_f is a fault state, s_f is a stressor influencing the fault growth, and F_{MPC} is the MPC-DDP controller from Module 1. Equation 3.48 and 3.49 show that MC is highly dependent on the reconfigured control policy, $\overrightarrow{\rho_R}$.

Figure 3.15 depicts an example of conceptual system performance propagation for different $\overrightarrow{\rho_R}$. The bold blue curve starting from t_0 to t_k is a trajectory traveled. t_k is a current

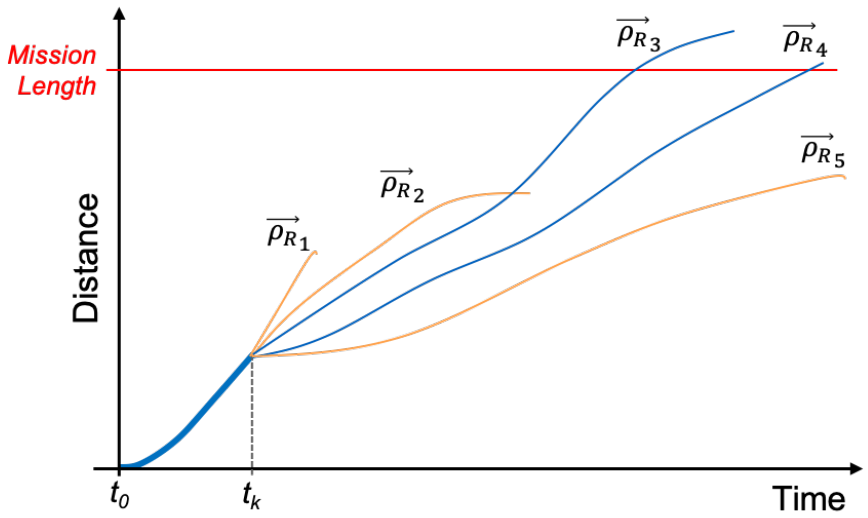

Figure 3.15: An example of conceptual system performance propagation (distance) for $\overrightarrow{\rho_R}$

time step. The series of the adaptation parameter of case 3, $\overrightarrow{\rho_{R3}}$, and case 4, $\overrightarrow{\rho_{R4}}$, will result in mission success, whereas the other three cases will not. The cause of three failures can be anything, not only the critical component failure. If the minimum mission time is preferable, $\overrightarrow{\rho_{R3}}$ will be a better choice than $\overrightarrow{\rho_{R4}}$.

Now, this is the same problem to DP in that solving this problem is to find the optimal series of the adaptation parameter, $\overrightarrow{\rho_R}$, within t_{k+N}. Therefore, the same challenges apply; t_{k+N} varies for different $\overrightarrow{\rho_R}$, and solving for a long time window is not practical due to the heavy computational burden.

One big difference from a typical DP problem is that the selection of $\overrightarrow{\rho_R}$ does not jeopardize the system stability and the trajectory-following performances much if ρ_R is within a safe range. Instead, $\overrightarrow{\rho_R}$ is more relevant to long-term mission capabilities such as mission length or mission time.

This difference, however, also causes another challenge. Selection of $\overrightarrow{\rho_R}$ with any combination of ρ_R at each t_{k+i} would result in different consequences. It implies that there are m^{N-1} combinations of $\overrightarrow{\rho_R}$ if $\rho_R \in \mathbb{R}^m$. With this regard, MC prediction for every single combination has a dimensionality problem. Moreover, N varies all the time based on x, x_f, and system noises. Therefore, modeling MC for every possible $\overrightarrow{\rho_R}$ is impossible.

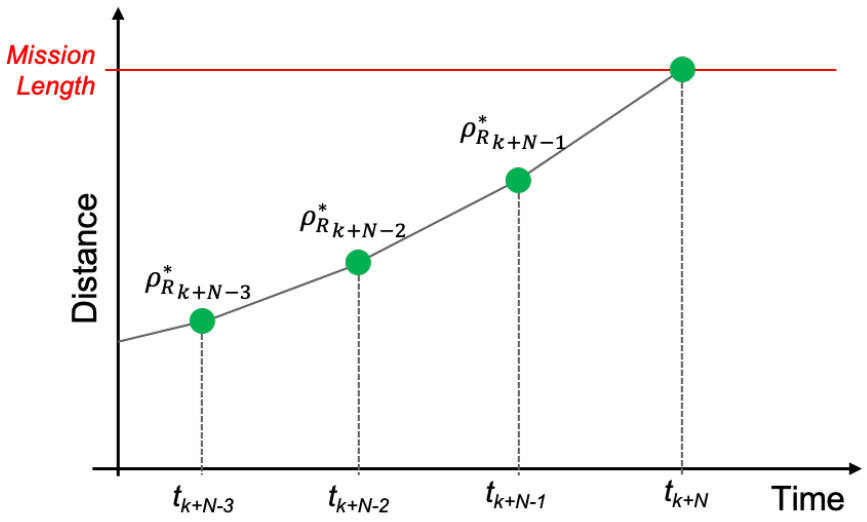

Figure 3.16: Conceptual representation of a series of optimal ρ_R

A simple and possible prediction modeling approach is to simplify the sequence of $\overrightarrow{\rho_R}$ by a constant ρ_R throughout the prediction period as shown below:

$$\overrightarrow{\rho_R} = \rho_R \times [1, 1, \cdots, 1]_{1\times N-1} \tag{3.50}$$

This simplification is a reasonable choice. Suppose that the series of optimal adaptation parameters, ρ^*_R, is known as depicted in Figure 3.16. Assuming a small time step size, states at consecutive time steps will be similar. It means that the optimal adaptation parameters at consecutive time steps will also be very similar.

$$x\left(t_i\right) \simeq x\left(t_{i+1}\right) \Rightarrow \rho^*_R\left(t_i\right) \simeq \rho^*_R\left(t_{i+1}\right) \tag{3.51}$$

A series of $\overrightarrow{\rho_R}$ with a constant ρ_R value will produce a certain consistent long-term mission capability with stable system stability and tracking performances. Figure 3.17 shows the example of conceptual system performance propagation with respect to different $\overrightarrow{\rho_{Ri}} = \rho_{Ri} \times [1, 1, \cdots, 1]_{1\times N_i}$ considering system noises. i is a case number. In this conceptual example, the average of both $MC\left(x\left(t_k\right), x_f\left(t_k\right), \overrightarrow{\rho_{R2}}\right)$ and $MC\left(x\left(t_k\right), x_f\left(t_k\right), \overrightarrow{\rho_{R3}}\right)$ exceed the mission length. $\overrightarrow{\rho_{R2}}$ finished the mission earlier than $\overrightarrow{\rho_{R3}}$. However, due to the

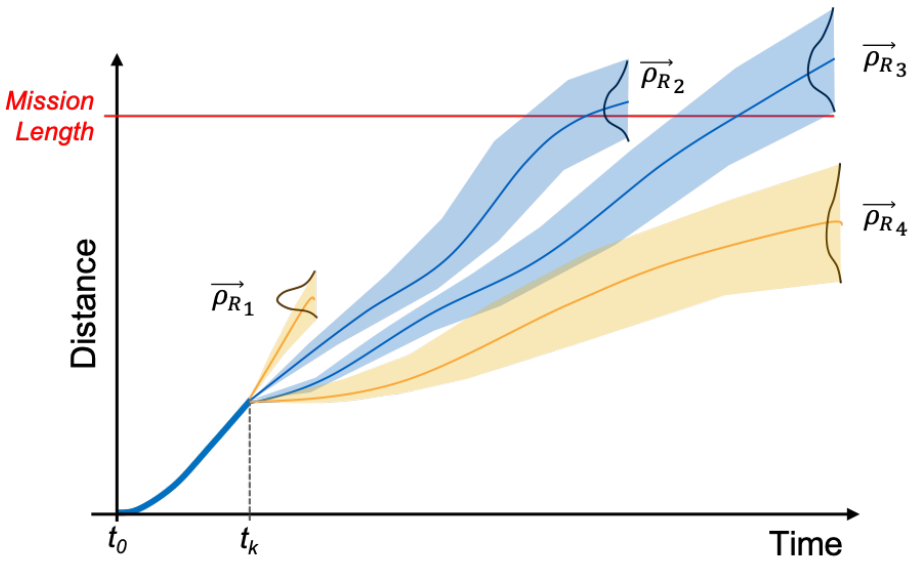

Figure 3.17: An example of conceptual system performance propagation (distance) with respect to $\overrightarrow{\rho_{Ri}} = \rho_{R_i} \times [1, 1, \cdots, 1]_{1\times N_i}$

uncertainty, the probability of success by $\overrightarrow{\rho_{R2}}$ is less than $\overrightarrow{\rho_{R3}}$. By certain statistical thresholds, most preferable $\overrightarrow{\rho_R}$ will be chosen. Mission time can also be estimated in the same manner as illustrated in Figure 3.18.

Now, $MC\left(x\left(t_k\right), x_f\left(t_k\right), \overrightarrow{\rho_{Ri}}\right)$ can be modeled in statistical parameters. For instance, by assuming a Gaussian distribution, MC prediction at t_k can be expressed as:

$$MC\left(x\left(t_k\right), x_f\left(t_k\right), \overrightarrow{\rho_{Ri}}\right) \sim N\left(\mu_i, \sigma_i\right) \tag{3.52}$$

By collecting MC for x, x_f, and $\overrightarrow{\rho_R}$ through multiple simulations, μ_{MC} and σ_{MC} can be modeled as shown in Eq. 3.53. Since $\overrightarrow{\rho_R}$ is a vector of the same value, simply ρ_R is used.

$$\begin{aligned} \mu_{MC} &= F_{mu_{MC}}\left(x, x_f, \rho_R\right) \\ \sigma_{MC} &= F_{\sigma_{MC}}\left(x, x_f, \rho_R\right) \end{aligned} \tag{3.53}$$

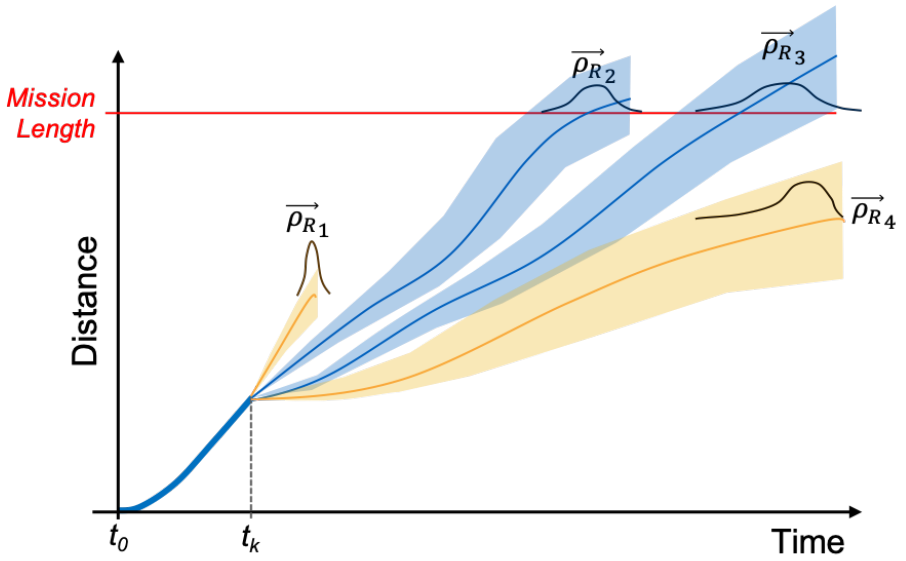

Figure 3.18: An example of conceptual system performance propagation (mission time) with respect to $\overrightarrow{\rho_{Ri}} = \rho_{Ri} \times [1, 1, \cdots, 1]_{1\times N_i}$

A statistical measure helps choose a robust long-term mission capability.

$$MC_\alpha = \mu_{MC} \pm z_\alpha \cdot \sigma_{MC} \tag{3.54}$$

where α is a confidence level, and z_α is a standard-z value for α. The desirability of MC determines the $\pm$ sign.

Now, the mapping function from x, x_f, and ρ_R to MC is established. The last part of Module 2 is to solve the optimization problem formulated initially in Eq. 3.5 and Eq. 3.6. Since MC can be directly predicted, a stationary $MC_{req.}$ can be chosen for x, x_f, or ρ_R.

The optimization problem is in a simple form, given the mapping functions. Any iterative solver would be applicable. If there is a clear trend, for instance, the objective function is monotonic in terms of ρ_R, then a zero-crossing detection algorithm would also solve for $MC_\alpha - MC_{req.} = 0$. Each time step, ρ_R is determined by the state estimation of x and x_f.

Alternatives Comparisons

Table 3.6 summarizes the characteristics of the two approaches proposed for Module 2. Their different characteristics made it difficult to identify which approach would work or be better than the other. Therefore, both approaches were tested. Test results are described in Chapter 4.

3.2.3 Module 3: Situational awareness

Assuming that a fault growth model is not known, the situational awareness module needs to identify two things:

1. Accurate fault detection and identification
2. Reasonable long-term mission capability prediction

As reviewed in Chapter 2, a particle filtering-based fault diagnosis approach is a state-of-the-art technique. For proper diagnosis performance, it requires a reasonably accurate fault growth model as well as data. By the assumption, however, a fault growth model is not known even though its typical patterns can be modeled. If an inaccurate fault growth model causes a significant delay or inaccuracy in fault detection time or state estimation, the system will not be as resilient as it should be. It formed Research Question 3-1 below.

Research Question 3-1

How to improve the performance of the particle filtering-based fault diagnosis routine when a fault growth model is unknown?

For the diagnosis performance improvement, online fault growth model estimation can be integrated into the particle filtering-based fault diagnosis routine. Thus, it led to Hy-

Table 3.6: A summary of characteristics of the two approaches

Reinforcement Learning (RL)	**Simulation-based Modeling**
• RL enables to learn a control strategy online while a system runs – Control strategies keep updated by new experiences. • RL does not require the knowledge of environments as long as states are formulated in the Markov Decision Process. • Producing an optimal solution at each control time can be nearly instant once an optimal control strategy is obtained. • It needs a huge size of training sets until a control policy converges because of the trial and error-based learning. • RL does not provide statistical measure of mission capability. – RL does not inform whether a faulty system can reach the destination or not. • Reward design is critical to learning performances, but rewards to not explicitly represent expected results.	• The Simulation-based long-term mission capability modeling approach can provide statistical measure of mission capability limits. • The simulation-based long-term mission capability modeling only predicts the capability envelop, not performance degradation over time. – Performance degradation and recovery over time are implicitly included in simulations. – Performance boundaries are imposed in simulations. • The simulation-based long-term mission capability modeling approach might be limited to apply to fast dynamics systems. – It needs to solve an optimization problem at each control step. • Once long-term mission capability models are built, it is challenging to update the prediction models online. – Offline modeling is required to incorporate new experiences.

pothesis 3-1 because the impact of model accuracy on diagnosis performance has not been shown:

Hypothesis 3-1

If online fault growth model estimation is introduced, performance of the particle filtering-based fault diagnosis will be improved when a fault growth pattern is unknown.

Uncertainty of a fault growth model makes it difficult to obtain an adequate control strategy for long-term mission capability. Different fault growth patterns yield different mission capabilities. Since Module 2 relies on missions and scenarios, it is challenging to model the mission capability for an unknown fault growth pattern as the fault growth model is estimated. It raises the need for a way to instantly predict the long-term mission capabilities for a new fault growth model.

Research Question 3-2

How to predict mission capability for Module 2 when a new fault growth model is identified?

The reasoning is a thinking process of making logical conclusions. The foundational elements are facts and premises. A broad scope of systems performing reasoning based upon facts and/ premises, or also called knowledge, are known as knowledge-Based Systems (KBS).

- **Definition: Knowledge-based systems (KBS)**

- A system that uses artificial intelligence techniques in problem-solving processes to support human decision-making, learning, and action from knowledge [88]

KBS is an architecture of a system, and there is not a consensus of generic methods of developing it [89]. Instead, KBS is system and domain-dependent. However, two standard components consist of KBS:

1. Inference engine
2. Knowledgebase

An inference engine is a computer software resembling decision processes of humans. It defines rules of reasoning about unknowns from the knowledgebase. The knowledgebase is a collection of knowledge, which represents facts about the world of interest. Knowledge can be represented by various forms for different applications: rules, semantic network, cases, frames, or decision diagrams. The choice of the form of knowledge should be determined to answer how humans (or experts) make decisions in similar situations [89].

The reasoning module of the proposed resilience-enhanced reconfigurable control framework is required to achieve the following purposes:

1. To detect and identify new events (faulty conditions) from the knowledgebase
2. To provide reconfiguration strategy to the long-term and short-term trade-off modules for known events
3. To suggest reconfiguration strategy to the long-term and short-term trade-off modules for unknown events
4. To store and update knowledge from new events

To serve the purposes, one of the most popular reasoning methods is Rule-Based Reasoning (RBR). RBR forms a set of rules representing critical characteristics of the domain

of interest. Rules consist of conditions and actions as multiple "If, then" statements. As a new event comes in, all or a sub-set of If-conditions are evaluated, and the individual actions are determined. Their rules are defined from expertise by domain experts. Even though RBR method has a history of successful development and application [90], practically it has limitations such as [91]:

1. A bottleneck problem in knowledge acquisition [92]: Expert knowledge is difficult to discover. Often, acquired rules do not accurately represent experts problem-solving procedures.

2. No memory for experiences: Only rules are stored, not experiences of both success and failure. Therefore, rules need to be evaluated whenever new events come in.

3. Lack of robustness: if no rules are matched, then no effective results are provided.

Case-Based Reasoning (CBR) is an alternative method for RBR. In this method, cases are forms of knowledge instead of rules. A case is a collection of information representing an exemplary experience. In general, a case is composed of information about (ref):

1. Problem

2. Solution

3. Outcome

A case, in this manner, can be constructed easier than expert rules because a problem statement in a case requires no expert knowledge, but only measurable features for each corresponding problem. These cases are updated in the case base as new experiences are encountered. With this regard, CBR is not just reasoning method, but also machine learning technique [68].

Considering the dynamic nature of a system, Procedural Reasoning System (PRS) reasons about procedural task planning, given states, goals, intention, and an act (or called

Table 3.7: Comparisons of reasoning methods

Properties	RBR	CBR	PRS
Expertise knowledge	High	Mid	High
Real-time applicability	Limited	Fair	Yes
Knowledge update	Difficult	Easy	Difficult
Memory required	Small	Moderate	Large
Integrability to Module 3	Good	Good	Bad

Knowledge Area, KA) library. PRS is known to apply to dynamical systems and real-time reasoning process in that it reactively provides partial and incremental task plans over time under a dynamically changing environment [93]. PRS is suitable for strategic planning when a mission consists of tasks in sequential order [94, 95]. A significant limitation of PRS is in difficulty of formulating KA [93].

Table 3.7 compares RBR, CBR, and PRS. From the comparison, CBR appears to be well-suitable for the problem of interest among alternative methods reviewed above.

Research Question 3-2

How to predict mission capability for Module 2 when a new fault growth model is identified?

Possible Answer

CBR is suitable for Module 3.

When a particle filtering-based fault diagnosis routine detects a fault, CBR needs to compare its growth pattern to cases in a case-base and eventually provide long-term mission capability-relevant knowledge to Module 2. It led to Hypothesis 3-2, as shown below. It will help prove the efficacy of CBR as Module 3 in the proposed framework.

Hypothesis 3-2

If a new fault growth model is detected, CBR will be able to produce reasonable mission capability prediction.

3.2.4 The integrated framework

Three modules described in the previous section are integrated as a sequential input-output relationship. Module 3 (CBR) takes a feature vector transformed from sensor measurements at each time. The parameter outputs and fault mode out of the CBR module are fed into Module 2. Module 2 determines the adaptation parameter given conditions. Lastly, Module 1 reconfigures control inputs to make the system satisfy long-term mission requirements and operate safely over time. Figure 3.19 illustrates the input-output relationship between the three modules.

The most fundamental and ultimate research question is if the system resilience can be improved through the proposed reconfigurable control framework.

Research Question 4-1

Will the proposed resilience-enhanced reconfigurable control framework improve system resilience?

Therefore, as an integrated framework, the proposed reconfigurable control framework needs to show the improvement of system resilience by an experiment. Chapter 2 presented different kinds of resilience metrics. In this study, the probability of mission success was chosen as the resilience metrics. It will be able to represent the improvement from a consequence perspective. It led to Hypothesis 4 below:

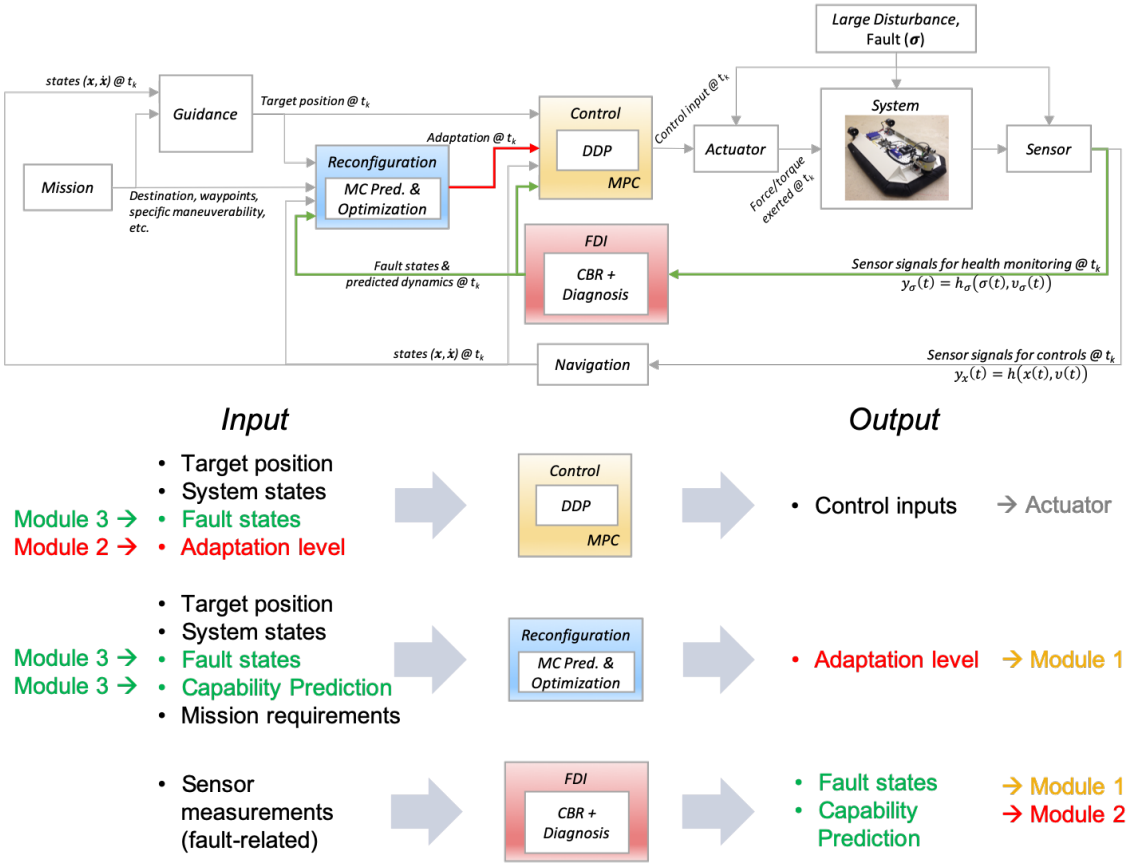

Figure 3.19: Input-output relationship of three modules in the resilience-enhanced reconfigurable control framework

Hypothesis 4

If the proposed resilience-enhanced reconfigurable control framework is applied, system resilience will be improved in terms of the probability of mission success in the presence of a critical fault.

Last, but not least, the proposed reconfigurable control framework needs to apply to a real application. For the real application, the whole process from monitoring and reasoning to MPC optimal control routine should be relatively fast.

Research Question 4-2

Will the proposed resilience-enhanced reconfigurable control framework be applied to real-time control applications?

It is challenging to come up with the computation time required for real-time application, though, because the real-time applicability highly depends on system dynamics, mission requirements, or fault growth rates. If dynamics is fast or a particular fault grows fast, then quick controls and responses are necessary. How quickly the proposed controller needs to respond is scenario-dependent. Due to this scenario-dependent issue, therefore, each computation time for each module was only observed during tests, instead.

3.3 Summary

Chapter 3 proposed a resilience-enhanced reconfigurable control framework for resilience improvement. Based upon a conventional AFTCS structure, three fundamental functional modules were introduced for the framework: immediate recovery, long-term mission ca-

pability recovery, and situational awareness. Research questions shaped the technical approaches, and preferable technical methods were reviewed, selected, and developed. Three modules are interconnected by input-output relationship. The proposed reconfigurable control framework is expected to improve the system usability in the presence of a critical fault mode, and eventually to enhance system resilience. Table 3.8 summarizes the proposed technical approaches, missing pieces that need to be shown, and hypotheses that help justify the proposed methods. Alternatives for the adaptation parameter optimization and the real-time applicability are directly tested and observed.

In Chapter 4, test cases and corresponding results are illustrated to support the hypotheses. As a testbed, a waypoint-following mission with an autonomously operable dynamical system, hovercraft, is introduced. Results and observations from a set of experiments show the efficacy of the proposed framework to satisfy the research objectives.

Table 3.8: Summary of the proposed technical approaches and hypotheses

Functions	Methods	Missing Information	Hypotheses
Immediate Performance Recovery	Optimization by MPC-DDP	Trajectory recovery capability	1. If MPC-DDP is used for Module 1, then the system trajectory will be recovered in the presence of a critical fault mode.
Long-term Mission Capability Recovery	Enabler: Adaptation parameter	The impact of an adaptation parameter on long-term mission capabilities	2. If the adaptation parameter varies, then long-term mission capabilities will be adjusted.
	Online optimization: RL vs. Simulation-based approach	Optimal solution for the adaptation parameter at each control time	
Situational Awareness	Online fault growth model estimation	The impact of model accuracy on diagnosis performance	3-1. If online fault growth model estimation is introduced, performance of the particle filtering-based fault diagnosis will be improved when a fault growth pattern is unknown.
	CBR	Prediction of long-term mission capability for a particular fault growth pattern identified	3-2. If a new fault growth model is detected, CBR will be able to produce reasonable mission capability prediction.
Integrated Framework		System resilience improvement	4. If the proposed resilience-enhanced reconfigurable control framework is applied, system resilience will be improved in terms of the probability of mission success in the presence of a critical fault.
		Real-time applicability	

CHAPTER 4
EXPERIMENTS AND RESULTS

Experiments need to show that the proposed technical methods are able to serve three main functions and eventually improve system resilience by the whole framework. For that, seven experiments are designed to support the hypotheses and fundamental functions, as summarized in Figure 4.1.

Experiment 1 supports Hypothesis 1. It is about how accurate fault knowledge would need for MPC-DDP to work correctly. Experiment 1, therefore, compared waypoint tracking performances of MPC-DDP with and without an accurate fault state estimation. An actuator fault mode was induced, causing a control asymmetry and a loss of controllability, in the waypoint-following mission. Experiment 1 is the foundation of the rest of the experiments.

Experiment 2 supports Hypothesis 2. It needs to prove the efficacy of the adaptation parameter for the long-term mission capability trade-offs. By introducing discrete levels of the adaptation parameter in the MPC-DDP objective function, long-term mission capabilities were observed via Monte Carlo simulations.

Experiment 3 is about the applicability of RL-based method for online optimization of the adaptation parameter. Learning behavior of RL and the quality of the learned control policy was evaluated in Experiment 3. Experiment 4, on the other hand, tested the applicability of the simulation-based long-term mission capability modeling and optimization method for the adaptation parameter. It turned out that the simulation-based method is suitable for the proposed framework, whereas the RL approach failed to show the applicability.

Experiment 1 through 4 were tested based on the assumption that fault states were estimated correctly. Fault state estimation, however, can be inaccurate. With a particle filtering-based fault diagnosis method, knowledge of fault growth dynamics is one of the

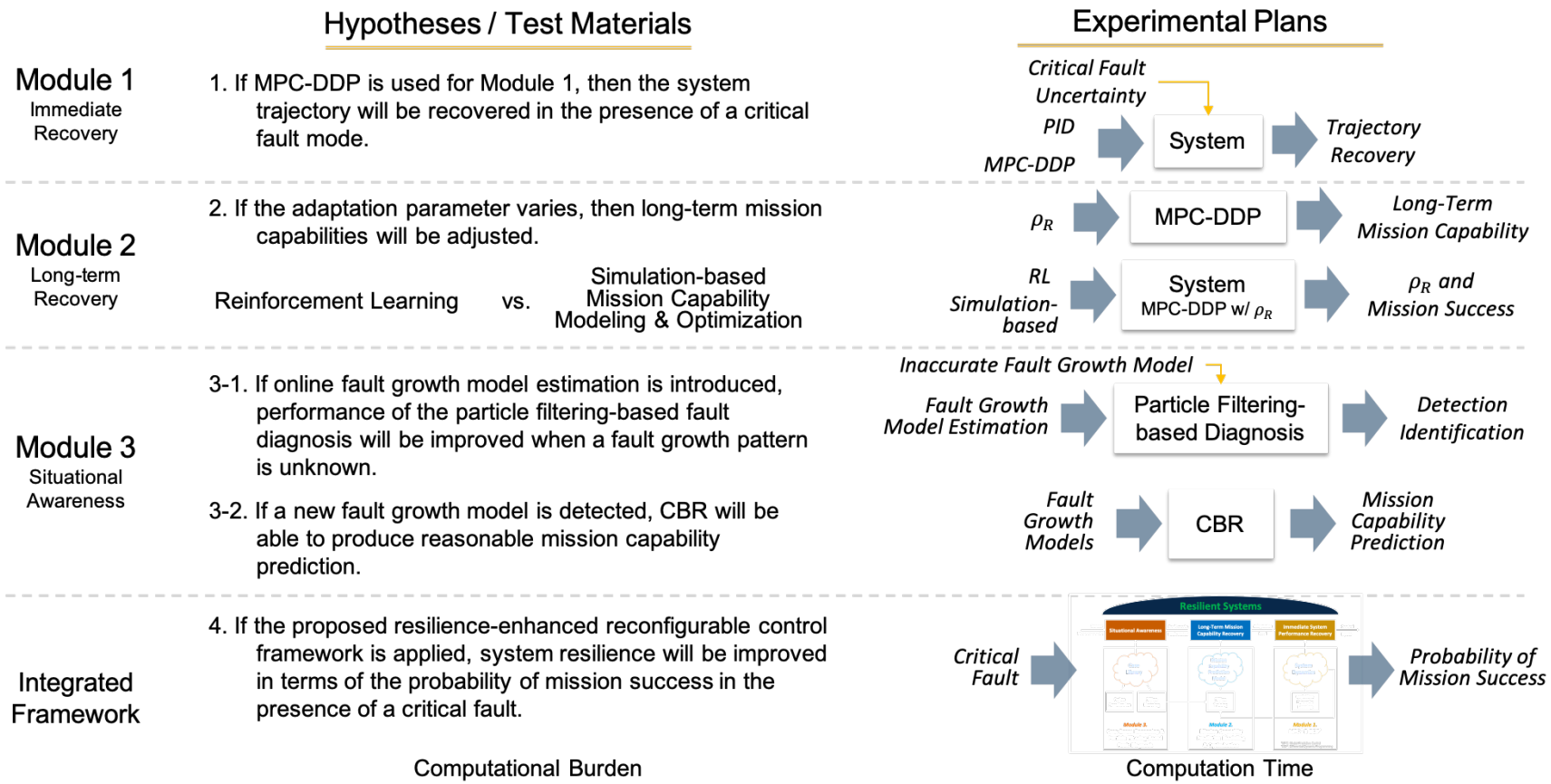

Figure 4.1: Hypotheses and experimental plans

uncertainty sources. RQ 3-1 posed a question whether the imperfection of the fault growth knowledge would impair the performances of fault detection and diagnosis. In Experiment 5, different fault growth models were tested in a particle filtering-based fault diagnosis module. Fault detection time was assessed. Fault states were estimated and compared to actual time and states.

Experiment 6 showed the applicability of CBR, which includes cases and a case-base for typical fault growth models. Fault growth characteristics were identified online, and a corresponding mission capability prediction model was used to optimize mission capability and to complete a mission. A new fault growth pattern was also introduced, and a solution transformation rule for CBR was evaluated by comparing mission capability estimated to simulation results.

Finally, Exeriment 7 evaluated the performance of the proposed resilience-enhanced reconfigurable control framework as one whole control system. The improvement of the probability of success was evaluated for the consequences of the framework. Also, the computation time for CBR, long-term mission capability optimization, and MPC-DDP were tracked to show the applicability of the proposed framework to a real application.

For such test plans, an under-actuated autonomously operable hovercraft was introduced as a testbed. Hovercraft operates on a 2-D plane, which makes it simple to develop a simulation environment. However, an under-actuated hovercraft can be challenging to control because of its sliding motions; therefore, it is significantly sensitive to actuator fault modes that are relevant to a system operating point. Without a doubt, reaching to the failure level of an actuator can be catastrophic, and low-level reconfiguration will not be able to handle the impact of the faults.

It does not necessarily mean that an under-actuated hovercraft is the only application for the proposed framework. It was only chosen to clearly show the impact of the fault and failure to the system operations.

This chapter starts from the descriptions of the testbed and missions. Then, the rest is

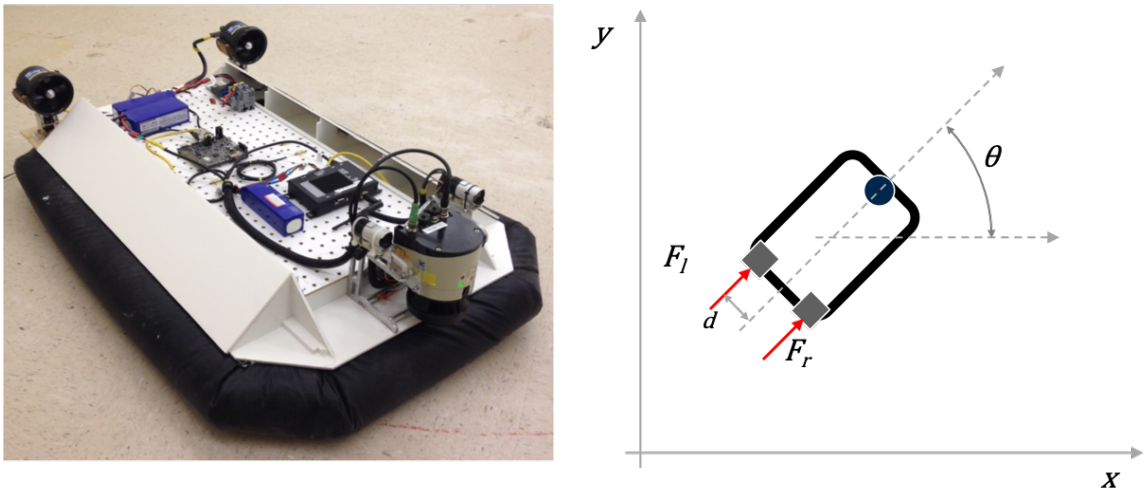

Figure 4.2: The autonomously operable hovercraft runs with two differential thrusts and a LIDAR sensor; The left figure shows the hardware built by ASDL at Georgia Institute of Technology, and the right figure represents hovercraft dynamics on a fix coordinate.

experiments and results. Each design of experiments was rationalized accordingly. Results supported hypotheses and finalized the proposed framework.

4.1 Testbed and Mission Descriptions

The Aerospace Systems Design Laboratory (ASDL) at Georgia Institute of Technology designed and built a scaled and autonomously operable hovercraft as a testbed, shown in the left figure of Figure 4.2 [96, 97]. The hovercraft dynamics model was derived based on a ground-fixed coordinate system as depicted in the right figure of Figure 4.2. The hovercraft operates with two differential thrust fans with electrical brushless motors and a LIDAR sensor for simultaneous localization and mapping. As a low-powered computing processor onboard, Pandaboard takes all signals with the help of Robot Operating System (ROS), which is a middleware handling sensor measurements and software signals.

The hardware platform, though, was only used to build a realistic software simulation environment in Matlab. Software simulations have benefits over hardware tests as described in [13]. Experiments can be expedited in a simulation environment, and it is easier and more flexible to induce artificial fault modes in simulations than hardware testing.

Table 4.1: Hovercraft System Properties

Parameters	Values	Descriptions
m (kg)	11.8	Vehicle mass
J (kg $\cdot$m^2)	1	Vehicle moment of inertia
d (m)	0.25	Distance between a thrust motor and longitudinal line crossing the mass center
d_t	0.5	Frictional damping (translation)
d_r	0.005	Frictional damping (rotation)

In the simulation environment, the hovercraft was assumed to move in two-dimensional planar motion space only with three degrees of freedom. Given two input controls, the hovercraft is under-actuated. Eq. 4.1 is the non-linear system dynamics model; x and y are absolute positions on the ground-fixed coordinate, θ is a heading angle, $\dot{x}$ is a velocity, $\ddot{x}$ is an acceleration, m is the mass, J is the moment of inertia of the hovercraft, and F_l and F_r are left and right thrust forces, respectively. d is the distance between a thruster and an imaginary longitudinal line crossing the mass center. It was assumed that the mass center coincides with the geometric center Based on the system dynamics equations, the state is $x = \left[x, y, \theta, \dot{x}, \dot{x}, \dot{\theta}\right]^T$, and the input is $u = [F_l, F_r]^T$. Table 4.1 shows the system properties used for the following experiments.

$$
\begin{aligned}
\ddot{x} &= -\frac{d_t}{m}\dot{x} + F_l \cdot \cos\theta + F_r \cdot \cos\theta \\
\ddot{y} &= -\frac{d_t}{m}\dot{y} + F_l \cdot \sin\theta + F_l \cdot \sin\theta \\
\ddot{\theta} &= -\frac{d_r}{J}\dot{\theta} + d\,(F_r - F_l)
\end{aligned}
\tag{4.1}
$$

The motor-fan dynamics was simplified as a typical DC motor-fan static model by assuming the dynamics of a motor is way faster than hovercraft dynamics. Figure 4.3 illustrates a typical DC motor structure. (past research) The DC motor model includes input voltage, V_{in}, armature current, I_m, armature resistance, R_m, and back electromotive force (emf) voltage, V_{emf}, as shown in Eq. 4.2.

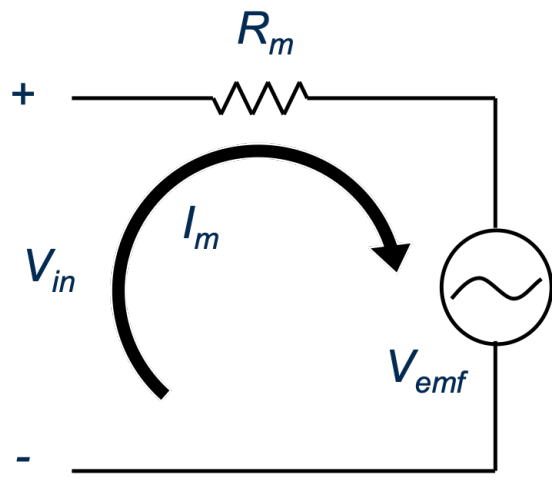

Figure 4.3: A typical DC motor structure

$$V_{in} = I_m R_m + V_{emf} \tag{4.2}$$

The back emf voltage is an induced voltage from the rotation of a rotor coil. The back emf voltage is represented by a proportional to an angular velocity of a rotor, Ω as shown in Eq. 4.3. A detailed derivation can be found from [98].

$$V_{emf} = k_e \Omega \tag{4.3}$$

where k_e is a back emf voltage constant.

A fan is a mechanical part producing torque. The torque equilibrium determines produced torque:

$$K_t I_m = b\Omega^2 \tag{4.4}$$

where $k_t I_m$ is produced torque, and $b\Omega^2$ is torque load from fan aerodynamics. k_t is a torque coefficient, and b is an aerodynamic load coefficient. The quadratic torque load model was assumed, and a simple hardware test verified it. Figure 4.4 shows the relationships between the angular speed of the rotor and the produced torque.

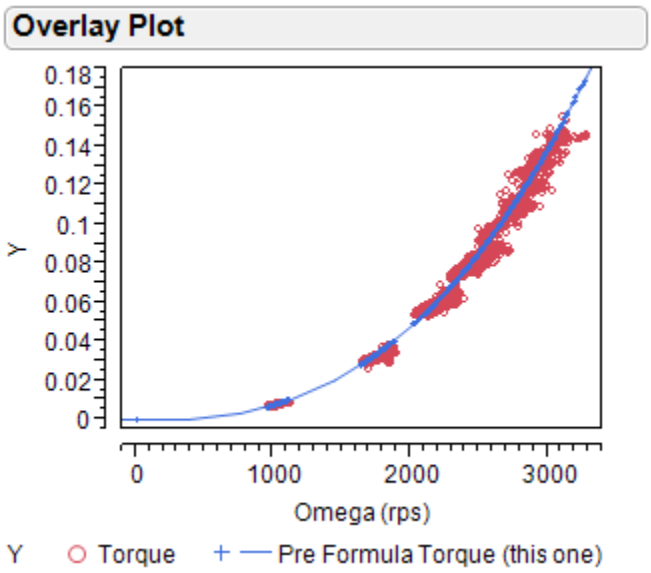

Figure 4.4: Test results of produced torque-rotor angular speed relationships

Table 4.2: DC Motor Model Coefficients

Coefficients	Values	Descriptions
k_e	0.006061893	Back emf coef.
k_t	0.006061893	Torque coef.
b	6.3867115e-9	Aerodynamic coef.
k_f	0.000000529025	Thrus force coef.

Thrust force exerted by the fan was also modeled in a quadratic form for the angular speed of the rotor as shown in Eq. 4.5.

$$F = k_f \Omega^2 \tag{4.5}$$

where k_f is a thrust force coefficient.

Table 4.2 includes the coefficients of the DC motor test model. The test model can produce up to 2.036 N on a maximum thrust effort.

A simulation environment can emulate four different uncertainty sources in general. The first uncertainty is aleatoric uncertainty. Equation 4.6 is state-space hovercraft dynamics with Gaussian noises with zero means and small variances. Variances were chosen to

disturb the system behavior enough, but not too large to make the hovercraft uncontrollable at all.

$$\begin{aligned}
\ddot{x} &= -\frac{d_t}{m}\dot{x} + F_l \cdot \cos\theta + F_r \cdot \cos\theta + \omega_x \\
\ddot{y} &= -\frac{d_t}{m}\dot{y} + F_l \cdot \sin\theta + F_l \cdot \sin\theta + \omega_y \\
\ddot{\theta} &= -\frac{d_r}{J}\dot{\theta} + d\,(F_r - F_l) + \omega_\theta
\end{aligned} \tag{4.6}$$

where

$$\begin{aligned}
\omega_x &\sim N(0, 0.013) \\
\omega_y &\sim N(0, 0.013) \\
\omega_\theta &\sim N(0, 0.013)
\end{aligned} \tag{4.7}$$

The second source of uncertainty is measurement noise. State estimation module in the simulation adds Gaussian noises with zero means and small variances.

$$\begin{aligned}
\tilde{x} &= x + \nu_x, \quad \nu_x \sim N(0, 0.05) \\
\tilde{y} &= y + \nu_y, \quad \nu_y \sim N(0, 0.05) \\
\tilde{\theta} &= \theta + \nu_\theta, \quad \nu_\theta \sim N(0, 0.03)
\end{aligned} \tag{4.8}$$

Numerical errors also add up uncertainties when velocity is estimated. A simple Euler's method was used. With the measurement noises,

$$\begin{aligned}
\tilde{\dot{x}} &= \frac{\tilde{x}_k - \tilde{x}_{k-1}}{dt} \\
\tilde{\dot{y}} &= \frac{\tilde{y}_k - \tilde{y}_{k-1}}{dt} \\
\tilde{\dot{\theta}} &= \frac{\tilde{\theta}_k - \tilde{\theta}_{k-1}}{dt}
\end{aligned} \tag{4.9}$$

where dt is a time step size, $t_k - t_{k-1}$.

Lastly, model errors in system dynamics can be a significant source of uncertainty. For instance, weight, the moment of inertia, lengths, or friction coefficients could be biased

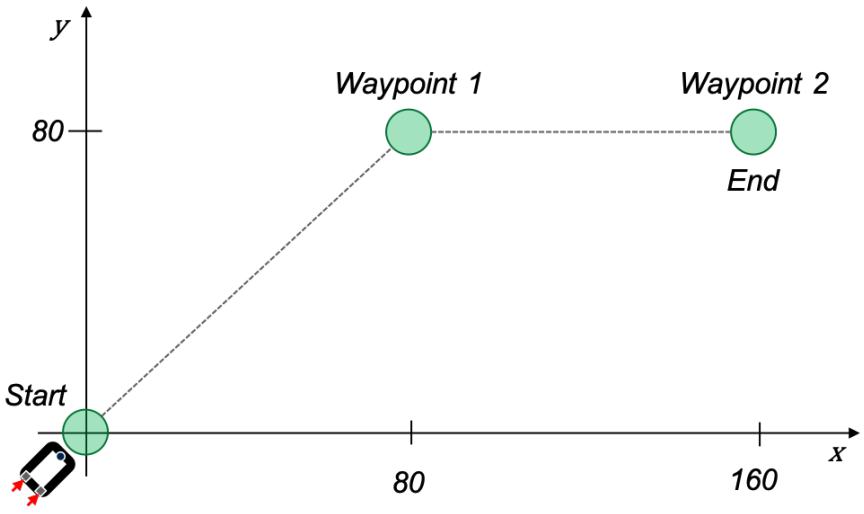

Figure 4.5: An example of a mission profile defined by waypoints

or inaccurate. Specifically, MPC-DDP requires accurate dynamics model to obtain proper optimal control inputs. Since the system dynamics is vital in MPC-DDP, this study assumes that accurate models are given. The impact of an inaccurate model was briefly investigated in Experiment 1, though.

The mission is simple. The hovercraft starts from a starting point, follows a series of waypoints, which are geographical positions, and ends the mission at the destination point. Mission complete time is a metric for mission performance. The goal of a mission is to reach the final destination as fast as possible.

The position vectors of waypoints define the mission profile. The waypoints were assumed to be given and known. Figure 4.5 depicts an example of waypoints on an x-y plane.

4.2 Fault and failure mode

Turn-to-turn short failure is one of the most common failure modes for electrical brushless motors [99, 100]. A complete short causes a total loss of thrust forces. Insulation degradation is the most common fault mode causing the turn-to-turn failure. Contaminants, abrasion, vibration, or voltage surge can cause insulation degradation. Once insulation

Figure 4.6: Turn-to-turn stator winding short of an electrical brushless motor [101]

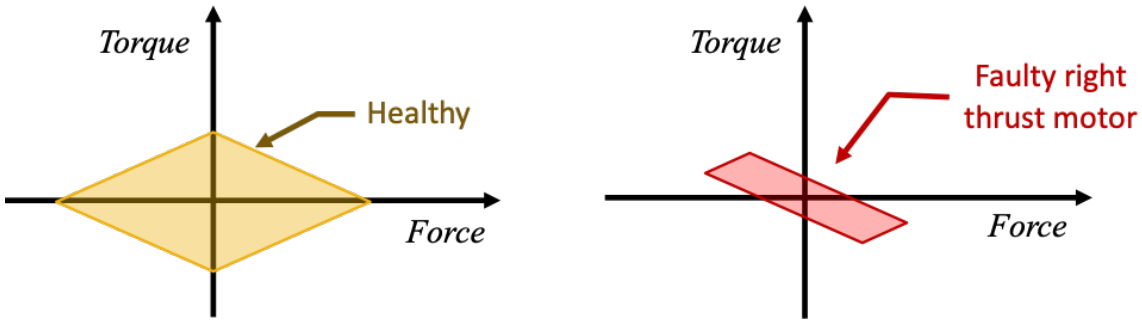

Figure 4.7: Notional comparisons of healthy and faulty hovercraft on torque-force plane: Left is a feasible torque-force envelop of healthy hovercraft, and right is a degraded envelop due to the control asymmetry.

degradation starts, the area of wing short expands. Figure 4.6 is a visible picture of turn-to-turn winding short [101].

A winding short leads to an increase in the resistance of a motor. High resistance implies the loss of motor efficiency and thrust force correspondingly. Impairment of one thrust motor, in turn, causes control asymmetry of hovercraft. Figure 4.7 depicts the impact of the control asymmetry by the feasible torque-force envelop from healthy (left) and faulty (right) hovercraft. Healthy hovercraft has a bigger feasible torque-force area than a faulty one. As the level of fault grows, the feasible area becomes shrunken more. It can be translated into the gradual loss of hovercraft control capability.

Fault growth is relevant to actuator loads [102]. In this study, loads were assumed to be

a function of control inputs. Without loss of generality, a fault growth rate was modeled as:

$$\dot{\sigma}(t) = f(t, S(t)) = f(t, u(t)) \tag{4.10}$$

where σ is a fault state, S is a load, and u is a control input.

A threshold in a fault state dimension defines failure. Once a severity level of a fault reaches the threshold, the component is no longer able to function; i.e., zero thrust force available.

$$F = 0, \qquad if \sigma \leq \sigma_{min} \quad or \quad \sigma \geq \sigma_{max} \tag{4.11}$$

where F is thrust force of hovercraft, σ_{max} and σ_{min} are maximum and minimum thresholds for failure.

Fault severity levels are usually difficult to measure online directly. The size of an armature winding short or the degree of insulation degradation requires overhauls of a motor and manual inspections. However, tracking armature resistance can represent the severity of the fault mode. By using the static equation of a DC motor, Eq. 4.2, 4.3, 4.4, and 4.5, armature resistance was estimated as derived in Eq. 4.12 and used as a fault feature in this study.

$$R_m(t) = \frac{V_{in}(t) - k_e \Omega(t)}{\Omega^2(t) \cdot \frac{b}{k_t}} \tag{4.12}$$

In the following experiments, fault states were represented differently for simplicity, though. Experiment 1 was enough to simulate only the impact of a gradual loss of thrust effectiveness only. Instead of an actual fault model, therefore, simplified fault effects were modeled as shown in Eq. 4.15.

$$F^{act} = \frac{F^{nom}}{\sigma} \tag{4.13}$$

where F^{act} is an actual thrust force exerted by the faulty thruster, and F^{nom} is a nominal thrust force.

By rewriting the Eq. 4.6,

$$\begin{aligned}
\ddot{x} &= -\frac{d_t}{m}\dot{x} + \frac{F_l^{nom}}{\sigma_l} \cdot \cos\theta + \frac{F_r^{nom}}{\sigma_r} \cdot \cos\theta + \omega_x \\
\ddot{y} &= -\frac{d_t}{m}\dot{y} + \frac{F_l^{nom}}{\sigma_l} \cdot \sin\theta + \frac{F_r^{nom}}{\sigma_r} \cdot \sin\theta + \omega_y \\
\ddot{\theta} &= -\frac{d_r}{J}\dot{\theta} + d\left(\frac{F_r^{nom}}{\sigma_r} - \frac{F_l^{nom}}{\sigma_l}\right) + \omega_\theta
\end{aligned} \tag{4.14}$$

Experiment 2 to 8 used a detailed model of an electrical motor to emulate fault dynamics and fault feature extractions. Therefore, the electrical motor-fan models, Eq. 4.2, 4.3, 4.4, and 4.5, were implemented in simulations, and armature resistance was monitored as a fault feature.

4.3 Experiment 1: Hypothesis test for Module 1 (MPC-DDP)

Experiment 1 was designed to test Hypothesis 1 and answer Research Question 1. By recalling Research Question 1 and Hypothesis 1,

- **Research Question 1**: What is a proper reconfigurable control method for Module 1? $\rightarrow$ MPC-DDP
- **Hypothesis 1**: If MPC-DDP is used for Module 1, then the system trajectory will be recovered in the presence of a critical fault mode.

If Experiment 1 shows that the hovercraft can manage its trajectory and proceed to the next waypoint in the presence of an electrical thrust motor insulation degradation, then it will support Hypothesis 1. Figure 4.8 describes the overall experimental procedure. After all, Experiment 1 needs to identify if a fault state estimation is necessary in order to run MPC-DDP properly as a reconfigurable controller. Thus, MPC-DDP for the hovercraft

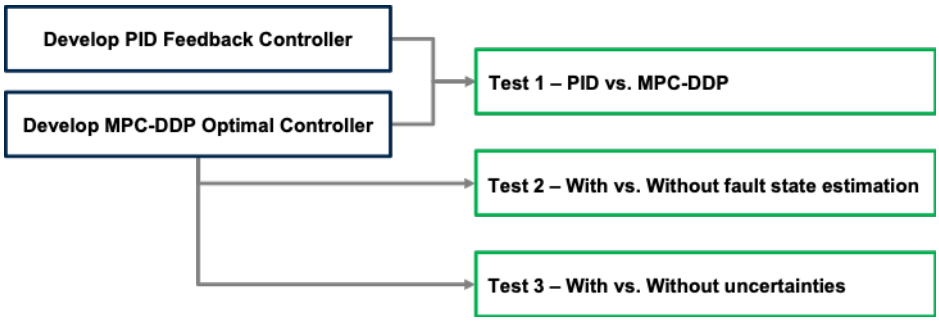

Figure 4.8: Overall procedure and test scenarios of Experiment 1

testbed was developed first, and its waypoint tracking performance as a fault tolerant controller was proved by comparing it to a traditional PID feedback controller (Test 1). Next, in Test Case 2, waypoint tracking performance was assessed with and without fault state estimation in the DDP formulation. To see the impact of various uncertainty sources, Test Case 3 was performed by inducing uncertainties in simulations.

4.3.1 Mission Description

In Experiment 1, two waypoints, including the final destination, were assigned in a mission profile as drawn in Figure 4.9. Starting from a point (0, 0), the hovercraft passes by the waypoint 1 (80, 80) and reach the final waypoint (160, 80). If the hovercraft got close to a waypoint less than a 3-meter radius, it was assumed that the waypoint was checked, and switched to the next waypoint.

4.3.2 Development of MPC-DDP Optimal Controller

As described in Chapter 3, the value function, V, at a time, t, of the optimal control-based controller was formulated in discrete time as Eq. 4.15.

$$V\left(x\left(t_k\right), t_k\right) = \min_{u}\left[\sum_{k=1}^{N} l\left(x\left(t_k\right), u\left(t_k\right), t_k\right)\Delta t + \Phi\left(t_N\right)\right] \tag{4.15}$$

System dynamics, Eq. 4.1 governs the state, $x = \left[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}\right]^T$. Control limits bound control input, $u = [F_l, F_r]^T$. Table 4.3 shows control limits used in the following

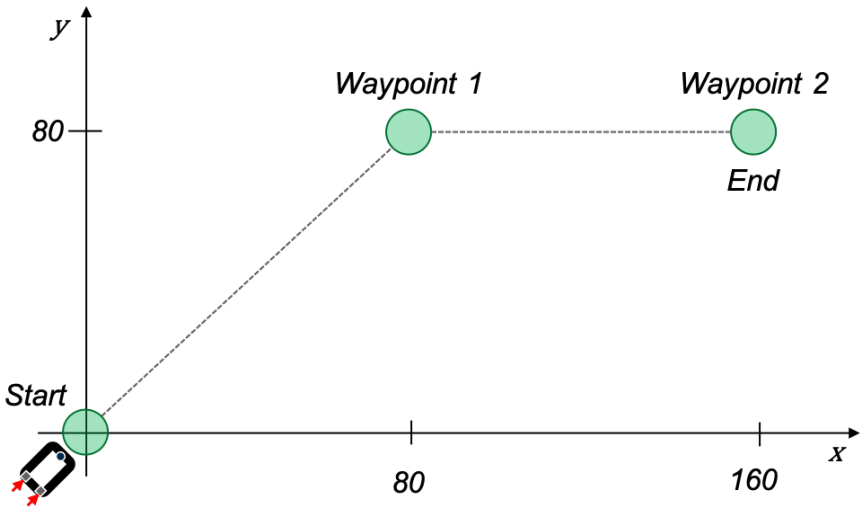

Figure 4.9: Mission profile for the test case (Two waypoints)

Table 4.3: Control Limits

Parameters	Values (N)
u_{min}	-2
u_{max}	2

tests, and Figure 4.10 depicts the corresponding torque-force envelop. A discrete time step, Δt, was set to 0.5 seconds. A receding horizon time window, t_N, was set to 10 seconds.

$$u_{min} \leq u \leq u_{max} \tag{4.16}$$

A running cost, $l\left(x\left(t_k\right), u\left(t_k\right), t_k\right)$, and final cost, $\Phi\left(t_N\right)$, were formulated in quadratic forms for the state, x.

$$l\left(x\left(t_k\right), u\left(t_k\right), t_k\right) = \frac{1}{2} x^T\left(t_k\right) K x\left(t_k\right) + \frac{1}{2} u^T\left(t_k\right) R u\left(t_k\right) \tag{4.17}$$

$$\Phi\left(t_N\right) = \frac{1}{2}\left(x\left(t_N\right) - x_f\right)^T K_f \left(x\left(t_N\right) - x_f\right) \tag{4.18}$$

The cost gains, K, R, and K_f, were manually tuned, and the final gains are shown in

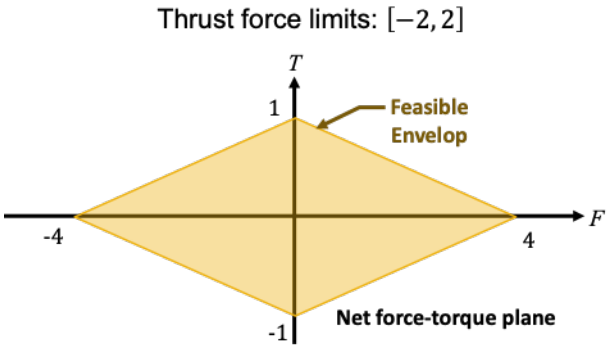

Figure 4.10: Torque-force envelop of the hovercraft with control limits, [-2, 2]

Eq. 4.19.

$$
\begin{aligned}
K &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix} \\
R &= \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \\
K_f &= \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{4.19}
$$

At each control time step, an optimal control sequence for a finite time can be obtained

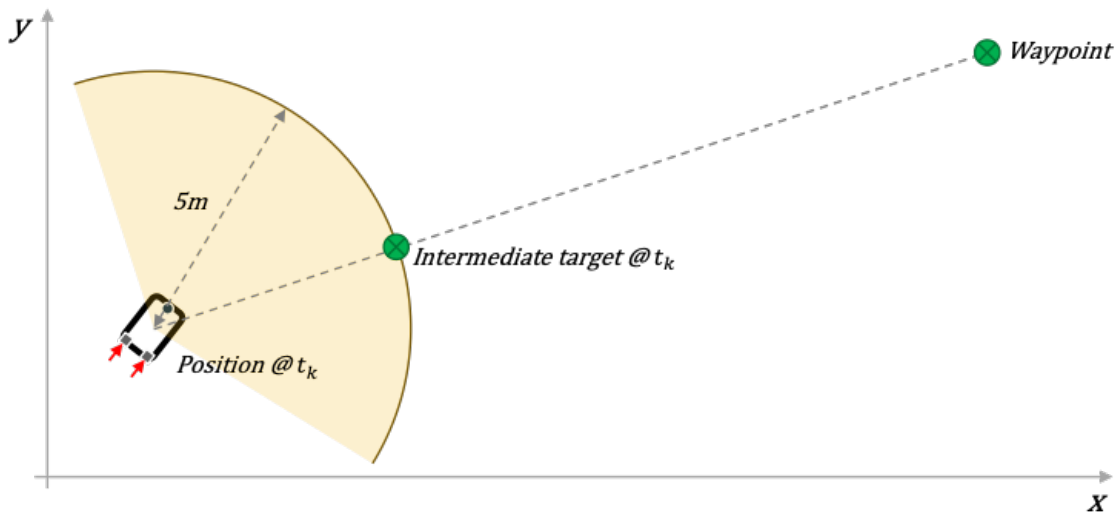

Figure 4.11: An intermediate target point for a finite time window in MPC

by repeatedly solving Eq. 4.20 until the control input, u, converges.

$$\delta u^* (t_k) = -Q_{uu}^{-1} Q_u - Q_{uu}^{-1} Q_{ux} \delta x (t_k) \tag{4.20}$$

To impose a steady cruise speed of hovercraft, an intermediate target at each control time was introduced. The intermediate target is a target position for each MPC time window. The intermediate target was determined by the position, 5 meters ahead from the hovercraft heading toward the waypoint. Since the MPC time window is 10 seconds, a reference for optimal control is to proceed 5 meters in 10 seconds. Figure 4.11 illustrates how to determine the intermediate targets at each control time.

4.3.3 Preliminary Results: MPC-DDP for normal (healthy) hovercraft control

In order to see if MPC-DDP work correctly for the hovercraft testbed, waypoint-tracking performances of nominal (healthy) hovercraft were first observed. Figure 4.12 shows the hovercraft trajectory following waypoints using MPC-DDP without uncertainty induced in simulation. Green circles are starting and waypoints. Little orange circles are position history of the hovercraft. Yellow arrows are heading directions of the hovercraft. The result

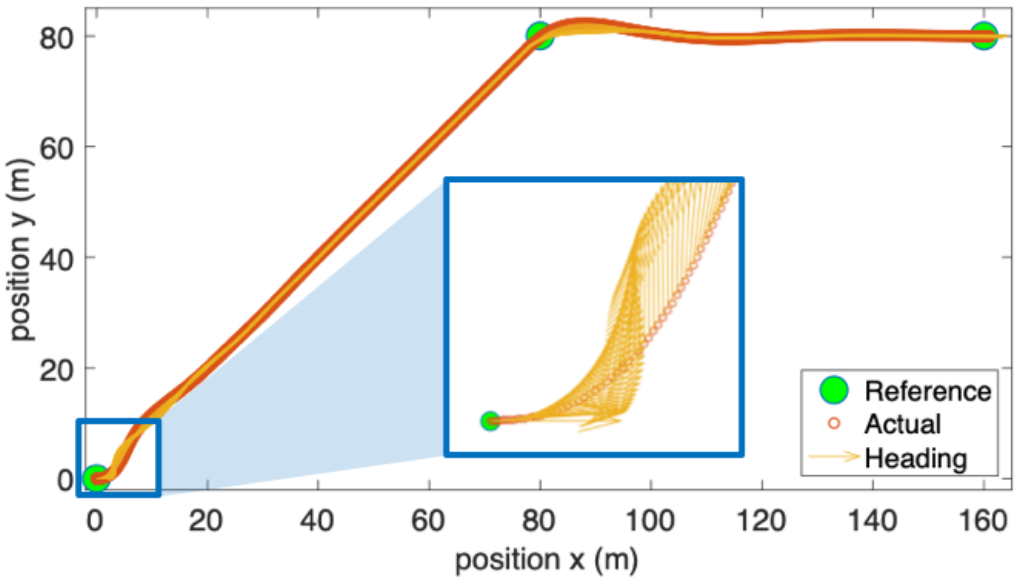

Figure 4.12: Results of waypoint-following trajectory and headings without noises

shows that MPC-DDP successfully found the optimal path (shortest route).

Figure 4.13 is the trajectory history with induced simulation uncertainties. The uncertainties included predefined aleatoric noises, measurement noises, and a numerical error. As expected, MPC-DDP successfully handled uncertainties and made the hovercraft reach to the final destination.

For both with uncertainty and without uncertainty cases, each state of the hovercraft was plotted in Figure 4.14. Red dashed lines represent state profiles without uncertainty, and solid blue lines show the results with uncertainties. The blue profiles fluctuate due to the induced uncertainties, whereas the red profiles are smooth. The uncertainty also caused delays. Without uncertainty, the mission ended at 472 seconds. With uncertainty, the mission time was 493.5 seconds.

Preliminary results showed the efficacy of MPC-DDP as an optimal controller for healthy hovercraft. Next set of tests induced an actuator fault in the middle of operations.

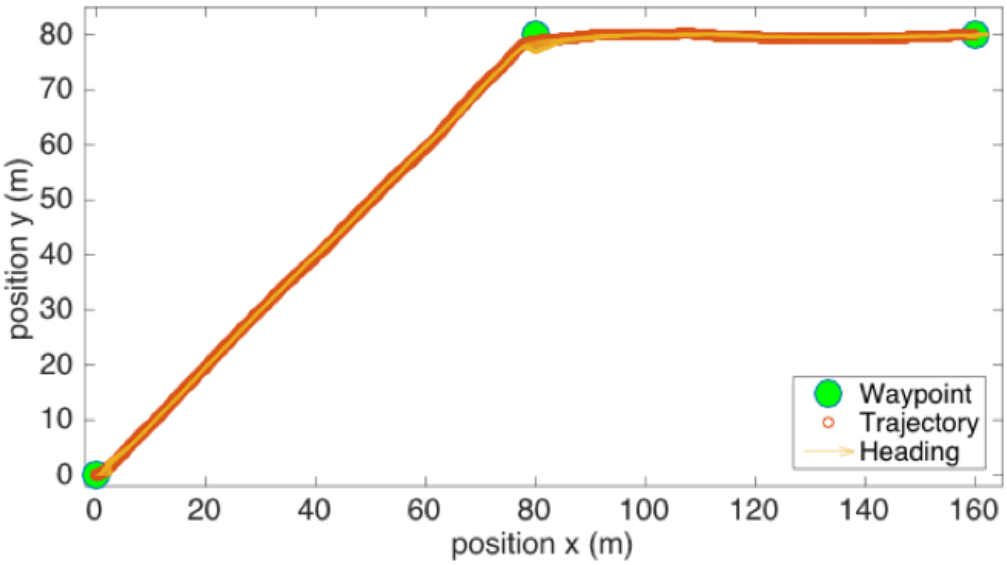

Figure 4.13: Results of waypoint-following trajectory and headings with noises

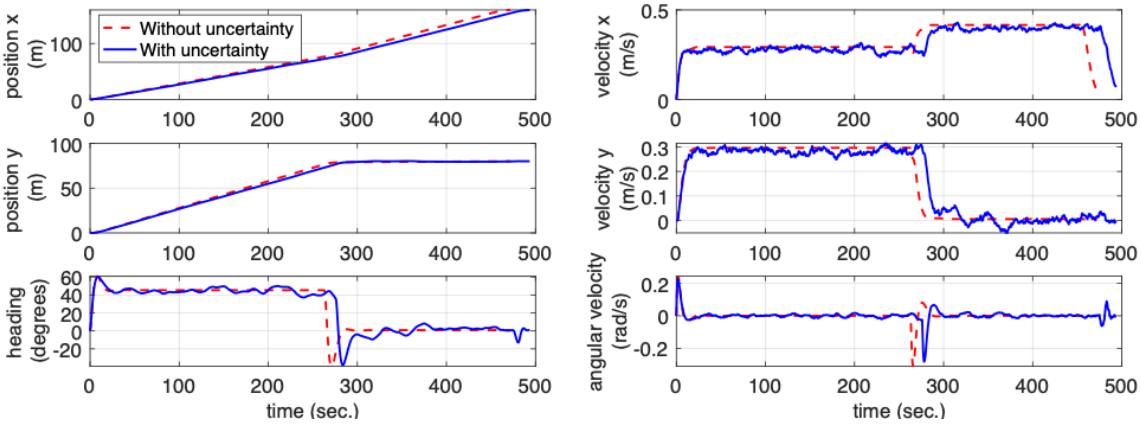

Figure 4.14: State profile comparisons: with noise (blue solid) vs. without noise (red dashed)

4.3.4 Test Case 1: An induced fault in the right thrust motor

Test Case 1 proves the fault tolerant control capability of MPC-DDP by comparing to a traditional feedback (PID) controller. A fault mode was induced on the right thrust motor starting at 80 seconds after the hovercraft was launched. The impact of the fault mode was modeled as shown in Eq. 4.21.

$$\begin{aligned}
\ddot{x} &= -\frac{d_t}{m}\dot{x} + F_l \cdot \cos\theta + \frac{F_r^{nom}}{\sigma_r} \cdot \cos\theta + \omega_x \\
\ddot{y} &= -\frac{d_t}{m}\dot{y} + F_l \cdot \sin\theta + \frac{F_r^{nom}}{\sigma_r} \cdot \sin\theta + \omega_y \\
\ddot{\theta} &= -\frac{d_r}{J}\dot{\theta} + d\left(\frac{F_r^{nom}}{\sigma_r} - F_l\right) + \omega_\theta
\end{aligned} \tag{4.21}$$

The fault growth rate was represented by a function of the control input, F_r^{nom}, as following:

$$\dot{\sigma}\left(t\right) = \rho_\sigma \cdot \left\{F_r^{nom}\left(t\right)\right\}^2 + \sigma_0 \tag{4.22}$$

where σ is a state of the fault, ρ_σ is a control-dependent coefficient, and σ_0 is a control-independent coefficient.

When the hovercraft is healthy, the fault state, , is set to one. Once the fault state reaches 10, then it is assumed that the thrust motor is failed, and no thrust force is exerted anymore.

A PID controller was able to successfully control the nominal (healthy) hovercraft as shown in Figure 4.15. Control gains were manually tuned. A reference forward surge speed was set to 0.45 [m/s] so that the average cruise speed of the PID-controlled hovercraft was close to the one controlled by MPC-DDP.

In a faulty-hovercraft scenario, however, the PID controller was not able to successfully control the faulty hovercraft and failed in the middle of the mission. Figure 4.16 demonstrates the mission failure of the hovercraft controlled by the traditional PID controller. Of course, it does not necessarily mean that a PID controller always cannot handle the effect

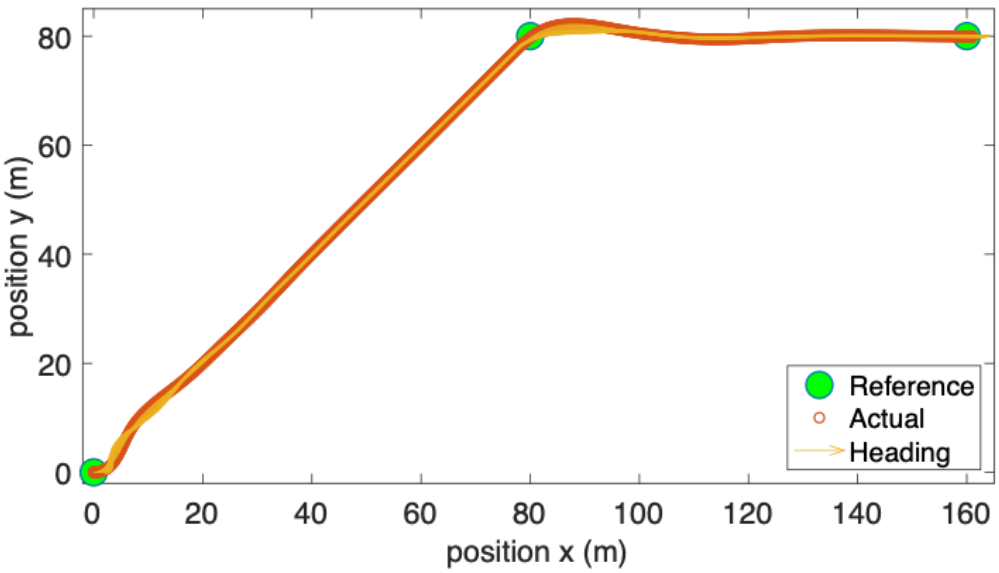

Figure 4.15: Hovercraft (nominal) trajectory controlled by a PID controller

of the fault. By the reconfiguration of state references or control gain rescheduling may be able to manage the fault effects and make the faulty hovercraft get to the final destination. It requires separate analyses for those reconfiguration rules.

In this test, a fault state was added to the system state vector as Eq. 4.24 and assumed to be accurate. Corresponding penalty coefficients in cost functions, K and K_f, were set to zeros.

$$x = \left[x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}\right]^T \tag{4.23}$$

Figure 4.17 shows an unusual trajectory controlled by MPC-DDP. After the first waypoint, the hovercraft oscillated back and forth, but still successfully managed to reach to the final waypoint.

The oscillating motion from MPC-DDP was, in fact, an optimal control strategy. As the fault level grows, actual thrust forces exerted from the faulty thrust motor decrease. To control the attitude (heading angle) of the hovercraft toward the waypoint, control efforts on the faulty motor needs to increase so that the actual thrust forces match to the desired

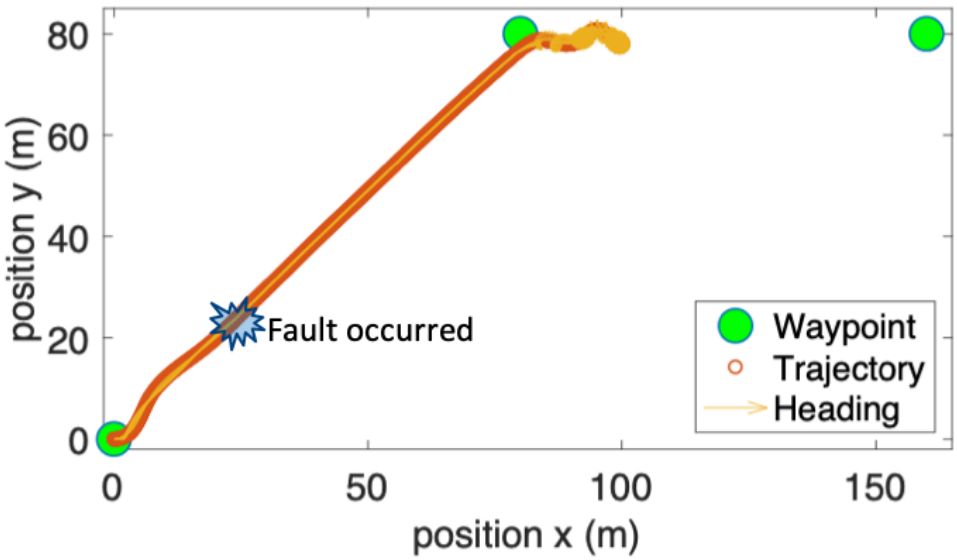

Figure 4.16: Failed mission trajectory of the faulty hovercraft controlled by the PID controller

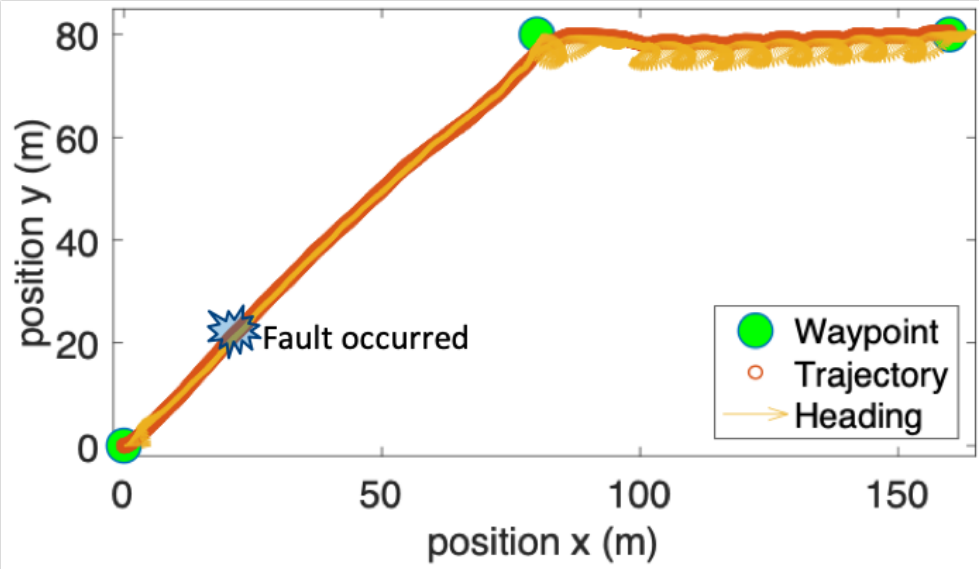

Figure 4.17: Successful mission trajectory of the hovercraft controlled by MPC-DDP

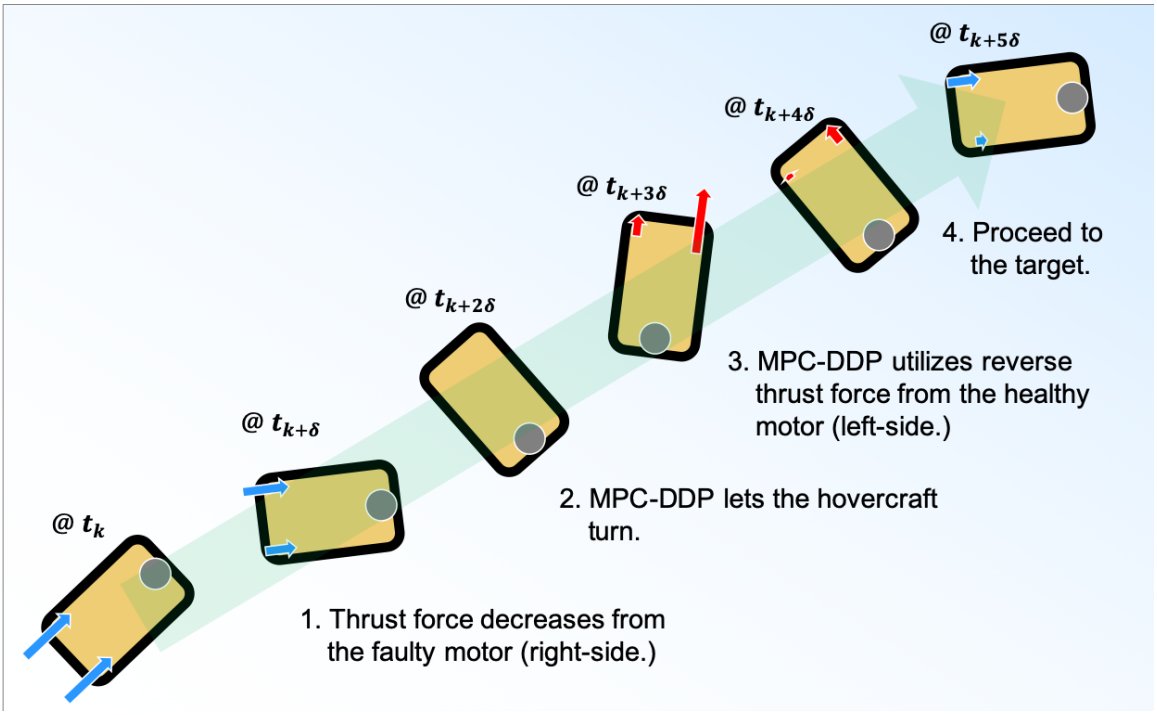

Figure 4.18: Demonstration of the derived control strategy by MPC-DDP for faulty hovercraft

thrust forces. The PID controller behaved this way, and eventually, it could not make the hovercraft controllable when the fault level reached the failure. MPC-DDP, however, found the way to minimize the control efforts on the faulty component. Instead of increasing the control efforts on the faulty thrust motor, it let the hovercraft turn backward, and used reverse thrust forces at a right timing in order to make the hovercraft head toward back to the waypoint and proceed to the waypoint at the same time. To help the understanding, Figure 4.18 illustrates the notional control strategy and the corresponding hovercraft behavior. Figure 4.19 compares the growing patterns of the fault level between PID and MPC-DDP. The fault level from the PID controller reached the failure limit way faster than the MPC-DDP controller.

The oscillating motions by MPC-DDP for the faulty hovercraft in Test Case 1 were not expected. This unusual behavior is one of the advantages of DDP; i.e., DDP can produce unexpected optimal patterns of a control sequence because it generates optimal trajectories and control inputs simultaneously. Of course, it depends on how cost functions are de-

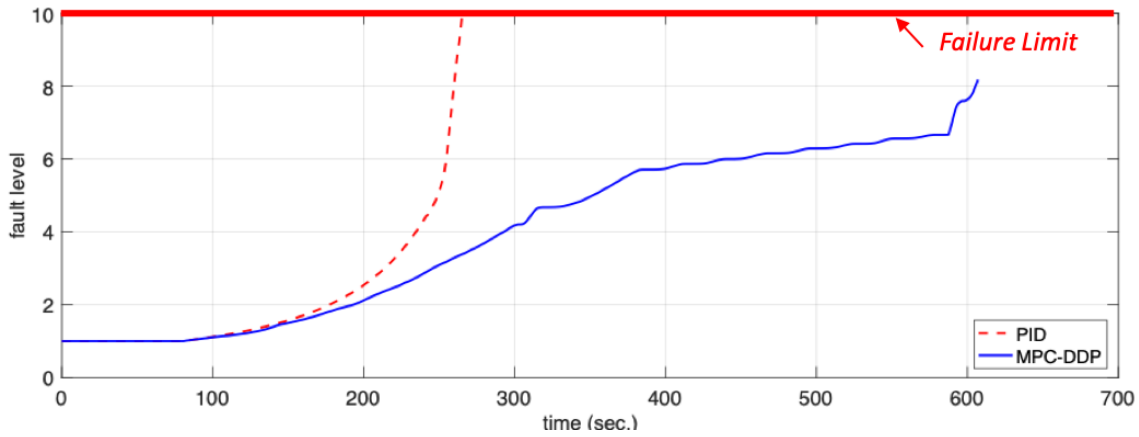

Figure 4.19: Fault growth pattern comparison: Red dashed line is PID, and blue solid line is MPC-DDP

signed. If heading angles were strictly forced to point the waypoint, the oscillating motions might not be shown.

The complete mission time for the faulty case was 605.5 seconds. It took around 120 seconds more than the healthy condition (preliminary test case).

4.3.5 Test Case 2: Impact of fault state knowledge in MPC-DDP

Based on the MPC-DDP design from Test Case 1, Test Case 2 omitted the fault state in the system state vector, x, and ran the same faulty condition as Test Case 1 in order to see if the fault state knowledge is necessary for the proper control reconfiguration. Interestingly, MPC-DDP without the fault state in the model was still able to make the faulty hovercraft to reach the destination as shown in Figure 4.20.

The hovercraft trajectory by MPC-DDP without the fault states was almost the same as MPC-DDP with fault states in the system dynamics. The hovercraft proceeded to the final destination with the oscillating motions. It turned out that MPC-DDP controlled the faulty hovercraft in the same way with a slightly different reason. This time, DDP did not plan to swing at first. Instead, the hovercraft turned and headed backward first due to the gradual loss of actual thrust forces. Once, it turned, DDP let the hovercraft swing and used the left thruster by reverse thrust because it yielded the less cost after all. Figure

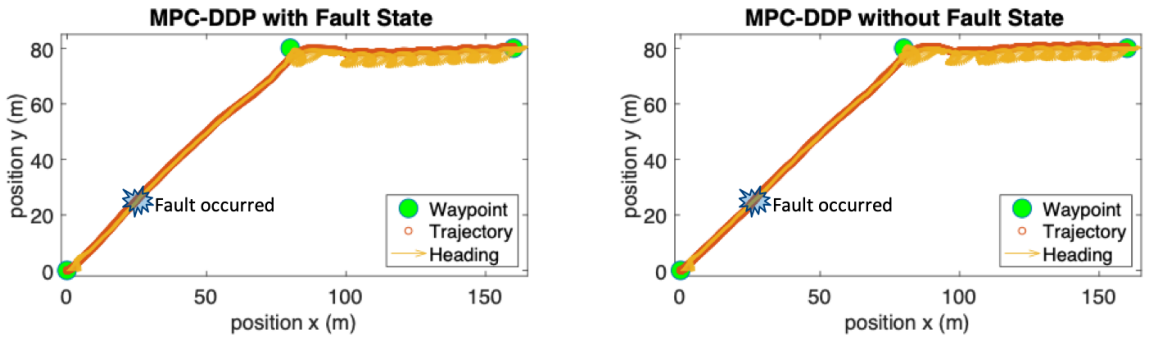

Figure 4.20: Trajectory comparison: MPC-DDP with the fault state knowledge (left) and MPC-DDP without the fault state knowledge (right)

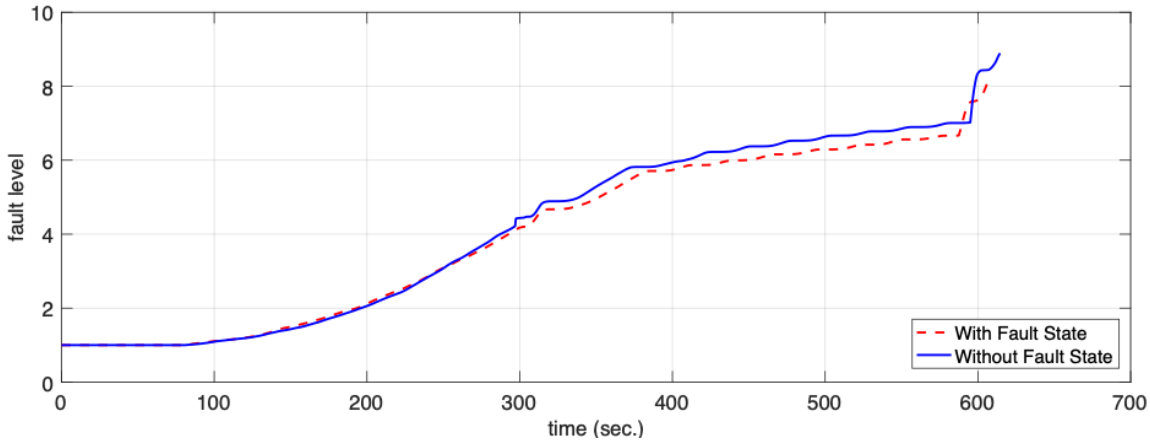

Figure 4.21: Fault growth pattern comparison: Red dashed line is the case with an accurate fault state estimation, and blue solid line is the case without the fault state estimation.

4.21 demonstrated the fault growth pattern difference between the case with the fault state knowledge (red dashed line) and without the fault state knowledge (solid blue line). The fault level grew more, and mission time took longer without the fault state knowledge, even though the difference was small.

The test results without the fault state knowledge suggest that MPC-DDP reconfigures control strategy and authorities flexibly as long as a system is controllable. According to Han and Zhao, under-actuated hovercraft is controllable if it can produce both positive and negative net torque. It means that the current testbed configuration is controllable even though one thrust motor fails if the other thrust motor can produce reverse thrust forces. Figure 4.22 depicts that net positive and negative torques are still feasible after the right

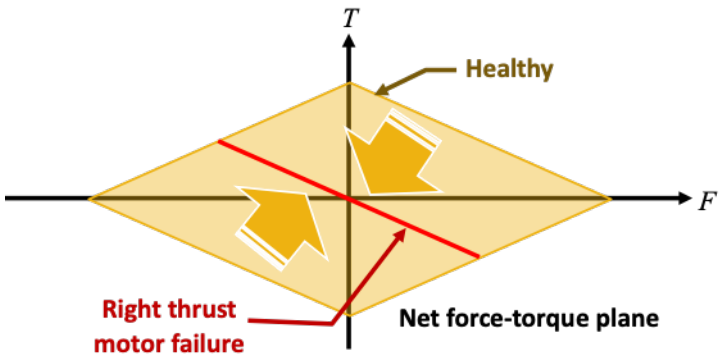

Figure 4.22: Feasible torque-force areas: healthy vs. right motor failure. The feasible area of the failed case collapses to a red line.

Table 4.4: Adjusted Control Limits

Parameters	Values (N)
u_{min}	0
u_{max}	2

motor completely loses its effective thrust forces.

Without reverse thrust forces, the hovercraft controllability decreases significantly. Table 4.4 is the adjusted control prohibiting reverse thrust force. Figure 4.23 is feasible torque-force areas for healthy and faulty conditions. When the right motor fails, the hovercraft becomes not controllable anymore due to the significant impairment in torque available.

The adjusted control limits changed the optimal control strategy entirely differently. No more oscillating motions happened. Without the fault state knowledge, MPC-DDP was not able to manage the faulty hovercraft to reach the final destination as shown in Figure 4.24.

With the fault knowledge, on the other hand, the mission length was extended. To effectively extend the mission length, the cost function in DDP included penalties by the level of the fault state. A certain coefficient setting elongated the mission length enough to complete the mission successfully. Figure 4.25 and Figure 4.26 show the trajectory of the successful case and fault level growth pattern, respectively. Without the fault states knowl-

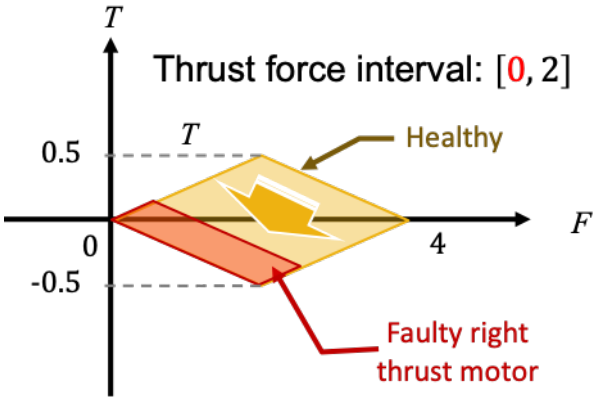

Figure 4.23: Feasible torque-force envelop for healthy and faulty conditions (no reverse thrust forces)

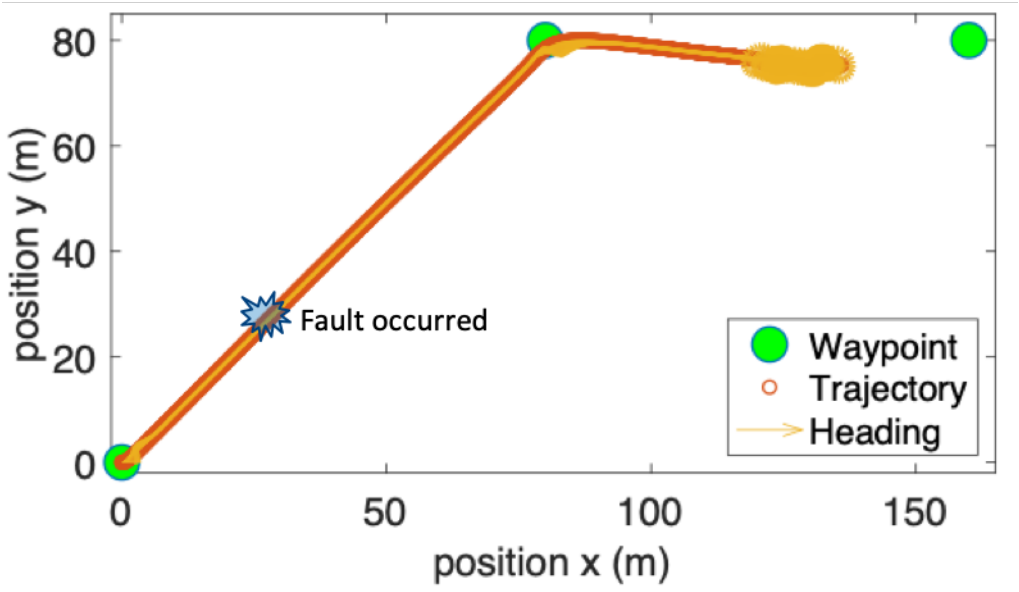

Figure 4.24: Mission failure by MPC-DDP without fault state knowledge (no reverse thrust forces)

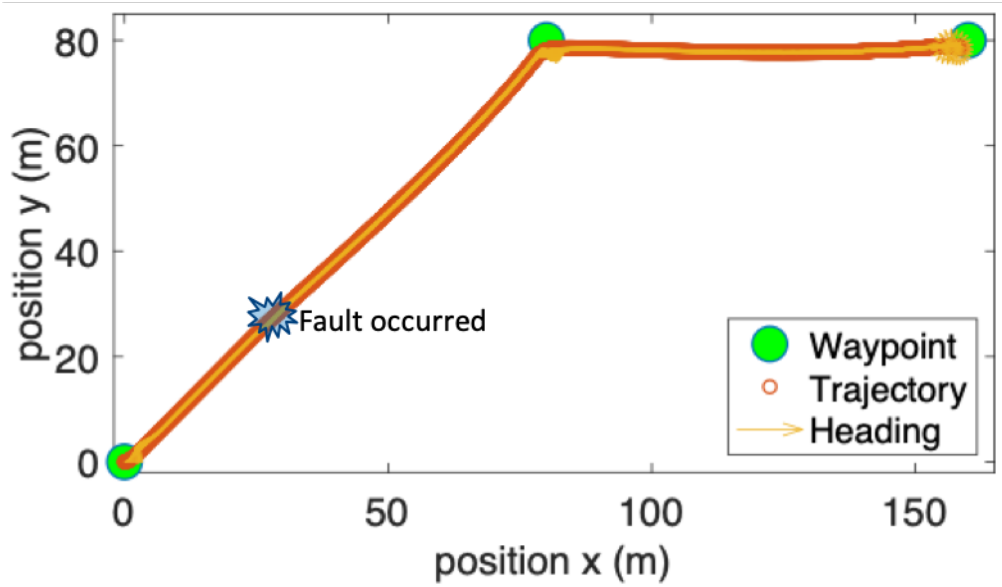

Figure 4.25: Mission success by MPC-DDP with fault state knowledge (no reverse thrust forces)

edge, the fault level grew fast and eventually became uncontrollable before the mission ended. With the fault states knowledge, fault growth was highly suppressed by minimizing the use of the faulty motor. Because of the small control efforts throughout the mission profile, the mission ended at around 1,360 seconds, almost four times longer than the nominal (healthy) case. It means that the control reconfiguration recovered the mission length at the expense of the mission time.

Note that the MPC-DDP control reconfiguration does not guarantee mission success. Design of the cost functions significantly affect an optimal control strategy. Certain cases could end up with mission failure even though the fault state is included in the DDP formulation.

In summary, Test Case 2 identified that the hovercraft configuration without reverse thrust forces is more vulnerable to the thrust motor fault mode with the MPC-DDP controller. The rest of the experiments from this point on used no reverse thrust force configuration for a severer scenario.

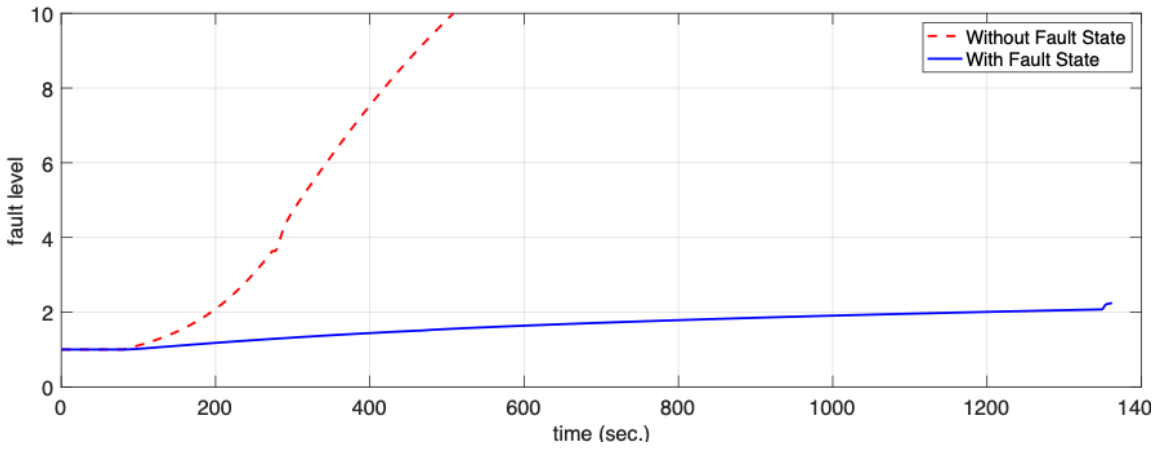

Figure 4.26: Fault level growth comparison (no reverse thrust forces): Red dashed line is without the fault state knowledge, and blue solid line is with the fault state knowledge.

Table 4.5: Inaccurate Model Properties

Parameters	Actual Values	Test Cases
m (kg)	11.8	10.8
J (kg) $\cdot m^2$)	1	1.2
d (m)	0.25	0.2
d_t	0.5	0.35
d_r	0.005	0.0035

4.3.6 Test Case 3: Impact of uncertainty sources other than fault states

The final test case of Experiment 1 was to investigate the impact of different uncertainty sources other than fault states knowledge. Test Case 3 assumed that the fault states are known. First of all, stochastic (aleatoric) system noises, measurement noises, and numerical errors were included all at once. The hovercraft controlled by MPC-DDP could reach the final target successfully as depicted in Figure 4.27.

However, inaccurate system property parameters caused severe performance degradation as shown in Figure 4.28. In the test case, 10 to 25 percent errors in property parameters were tried as summarized in Table 4.5.

Test Case 3 suggested that MPC-DDP properly managed a certain level of stochasticity of system dynamics, zero-mean measurement errors, and numerical errors. However,

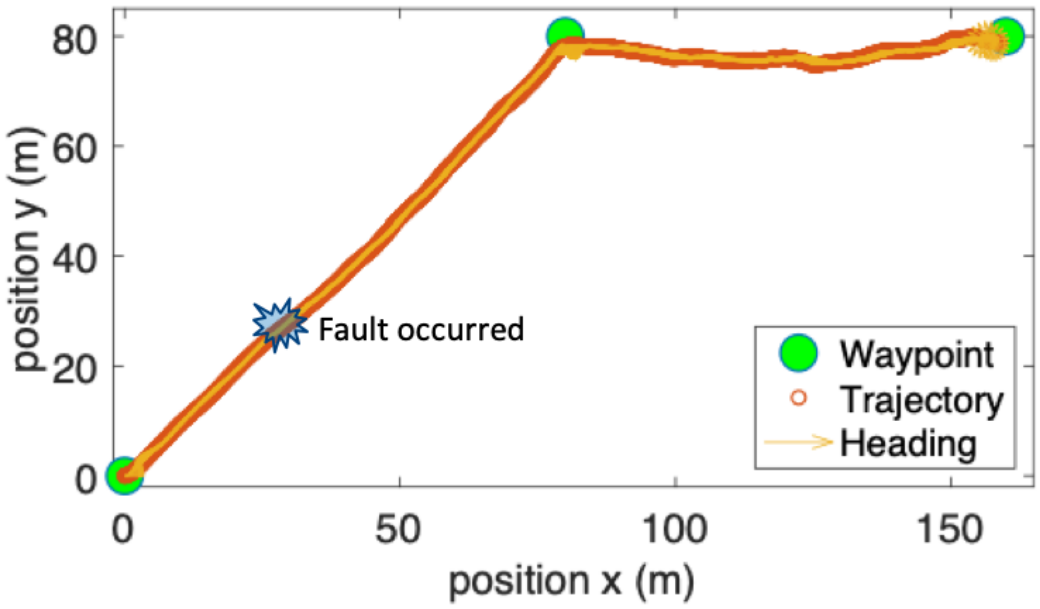

Figure 4.27: Hovercraft trajectory with aleatoric noises, measurement noises, and numerical errors

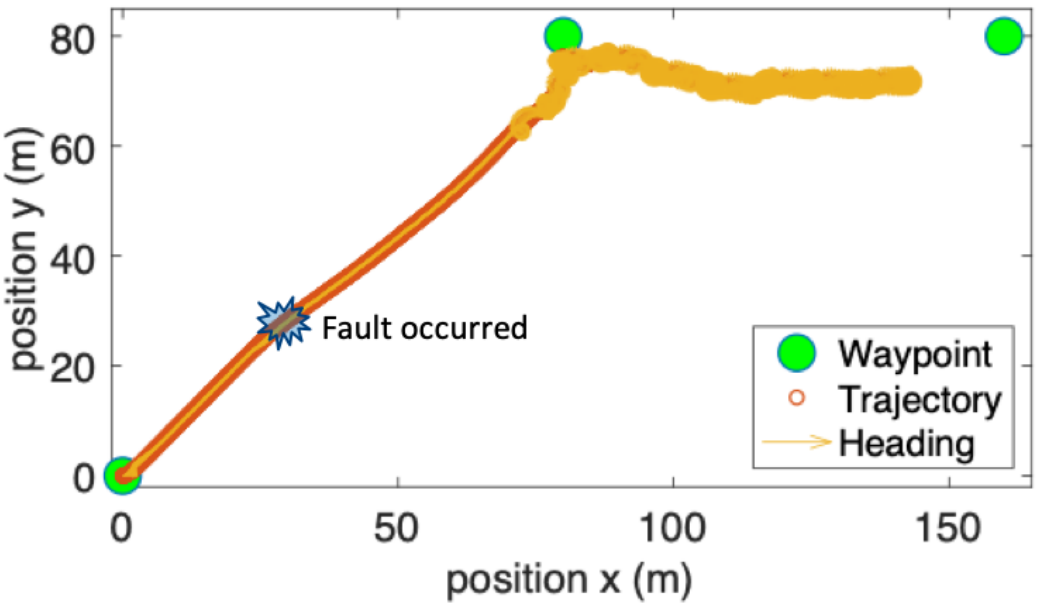

Figure 4.28: Hovercraft trajectory with inaccurate system properties

biases in model properties prevented MPC-DDP from producing optimal control strategies.

4.3.7 Summary of Experiment 1

Results of Experiment 1 supported Hypothesis 1 by showing that MPC-DDP enables trajectory recovery in the presence of a critical fault mode. For the tests, an MPC-DDP controller for the under-actuated hovercraft was developed and verified as a nominal and fault tolerant controller by comparing its control performances to a typical PID controller. In order to test Hypothesis 1 and answer Research Question 1, different sources of uncertainties were demonstrated in the existence of the thrust motor fault. Observations were summarized below.

- Observations

    1. A typical PID controller was not able to control a faulty hovercraft due to the fast growth of the fault.
    2. MPC-DDP seemed to find optimal paths and control inputs if the system controllability was not severely impaired.
    3. The design of cost function would significantly affect the optimal solutions.
    4. MPC-DDP required the knowledge of fault states when the fault effects were critical.
    5. MPC-DDP was able to manage stochastic noises in dynamics, zero-mean measurement errors, and numerical errors.
    6. MPC-DDP was NOT able to produce proper optimal solutions when the system property parameters were inaccurate.
    7. The mission length in the existence of the thrust motor fault was extended at the expense of the mission time.

4.4 Experiment 2: Hypothesis test for Module 2 (Adaptation parameter)

Experiment 2 was designed to test Hypothesis 2 and answer Research Question 2. By recalling Research Question 2 and Hypothesis 2,

- **Research Question 2**: How to obtain an optimal solution from Module 1 considering long-term mission capability? $\rightarrow$ By adaptation parameter
- **Hypothesis 2**: If the adaptation parameter varies, then long-term mission capabilities will be adjusted.

If Experiment 2 shows that long-term mission capabilities (mission time and mission length) are adjusted by varying the adaptation parameter, then it will support Hypothesis 2. It means that the adaptation parameter, ρ_R, can be a variable for long-term mission capability optimization.

The adaptation parameter determines the level of penalties in terms of control efforts. It was hypothesized that a small adaptation parameter leads to agile movements by an aggressive control strategy and that a high adaptation value leads to slow movements.

Experiment 2 introduced an adaptation parameter in the MPC-DDP controller developed in Experiment 1. By varying the adaptation parameter from 0.1 to 100, two mission capabilities were evaluated: the mission success/failure and the mission time. For simplicity, a single-waypoint mission was assumed. The total mission length was set to 80 meters, and the fault started randomly at any point in the middle of the mission. With uncertainties induced, Monte Carlo simulation was done for 10,000 test cases.

Figure 4.29 depicts the results regarding the mission length for the adaptation parameter. The horizontal axis is the adaptation parameter values that randomly chosen in each test case. The vertical axis is the mission lengths left when the faults occurred. For instance, data points drawn at the top area of the plot imply that faults began early in the mission, leaving long mission distances. The color code distinguishes between successful (blue) and

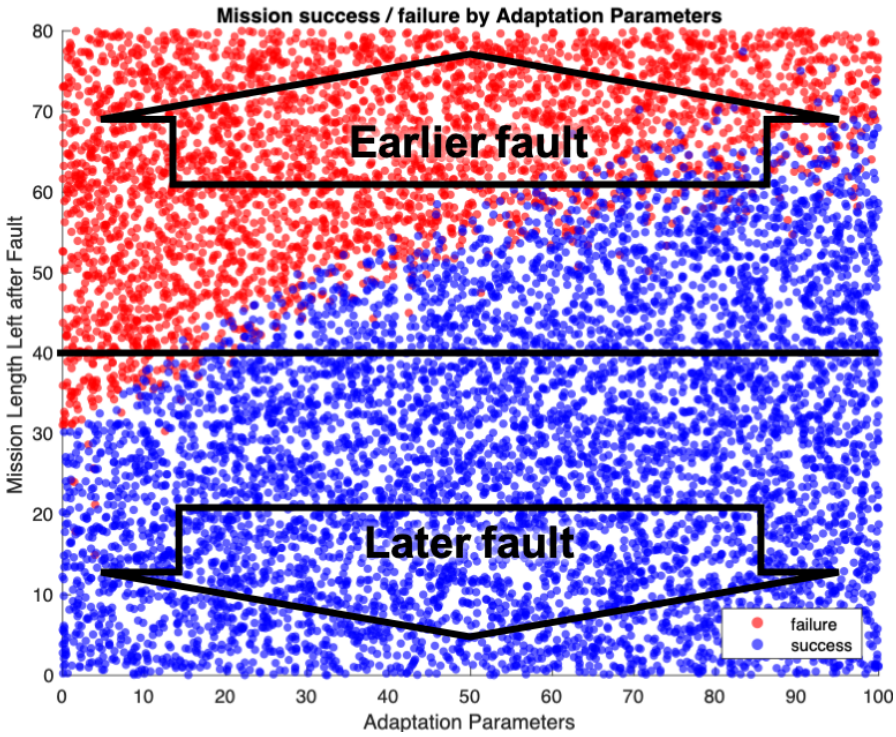

Figure 4.29: Adaptation parameter vs. mission length

failure (red) cases. The faulty hovercraft was not able to reach the final destination when the mission length left was long. Also, the higher the adaptation parameter value was, the longer the mission length was extended. With this plot, one can say that if 40 meters of the mission length is left when the thrust motor fault mode is detected, 30 or higher should be chosen for the adaptation parameter.

Note that mission success and failure was clearly distinguished at smaller adaptation values than greater ones. For instance, at around 80 of the adaptation value and 65 of the mission length left after fault, there seem to be success cases just as much as failure cases. It means that mission lengths were prone to uncertainties at larger adaptation values.

According to Experiment 1, a longer mission time was inevitable to extend the mission length. Figure 4.30 shows the same pattern as Experiment 1. The larger the adaptation parameter value was chosen, the more the mission time was required. The color-coding legend means where the fault occurred during the mission. For example, blue dots are cases where the fault occurred early (0-16 meters away from the starting point) in the mission.

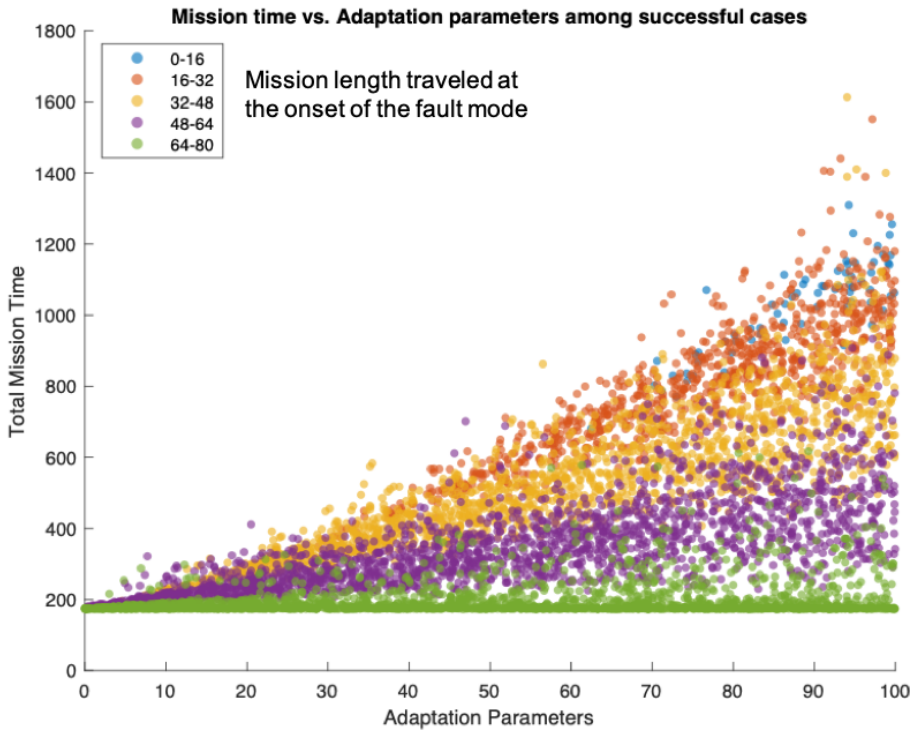

Figure 4.30: Adaptation parameter vs. mission time

They are cases where the adaptation parameter values affect the most.

Figure 4.30 also showed higher variances at large adaptation values than low values.

4.4.1 Summary of Experiment 2

Results of Experiment 2 supported Hypothesis 2. It showed that the proposed adaptation parameter results in long-term mission capability adjustment. In the hovercraft test scenario, mission success/failure (mission length) and mission time were observed by inducing the thrust motor fault mode at any point in the waypoint following mission. In order to test Hypothesis 2, different levels of the adaptation parameter were tried once the fault mode was initiated. Observations were summarized below.

- Observations

    1. Faults at an earlier mission phase more likely caused mission failure due to a higher mission length left.

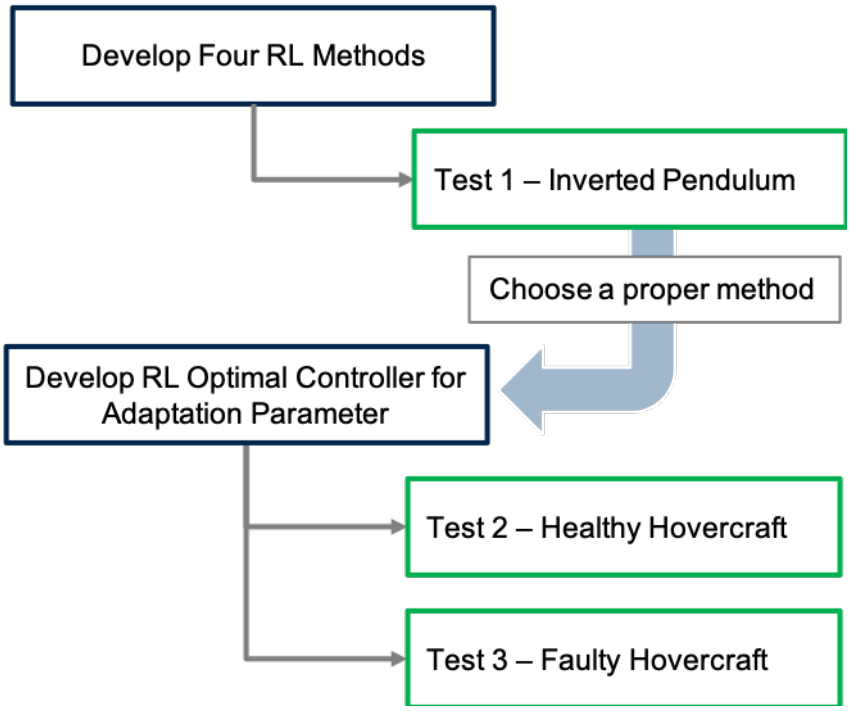

Figure 4.31: Overall procedure and test scenarios of Experiment 3

2. Larger values of the adaptation parameter increased the rate of mission success.
3. Larger values of the adaptation parameter increased the mission time.
4. Mission performances were prone to uncertainties at large adaptation values.

4.5 Experiment 3: Reinforcement learning for adaptation parameter optimization

If Experiment 3 shows that RL finds an optimal control strategy in terms of ρ_R, then RL can be a candidate as a method of Module 2. Thus, Experiment 3 was designed to examine the efficacy of RL for Module 2 in the proposed resilience-enhanced reconfigurable control framework.

Figure 4.31 depicts the test procedure of Experiment 3. First, before the RL approach was tested in the hovercraft example, four different RL methods were tested in the inverted pendulum example in order to examine the characteristics of each RL method. The four RL methods were:

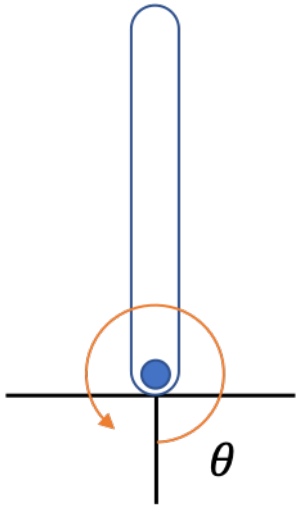

Figure 4.32: Inverted pendulum example

1. Q-learning / SARSA learning
2. Deep Q-Network (DQN) using Neural Network value function approximation
3. Double DQN (DDQN)
4. Deep Deterministic Policy Gradient (DDPG) using NN approximation for both value and policy (Actor-Critic method)

Based on Test 1, one proper method was chosen and modified for the adaptation parameter within the proposed resilience-enhanced reconfigurable control framework. Test 2 evaluated the learning behavior of RL and waypoint-following performance of healthy hovercraft. Finally, the right thrust motor fault mode was introduced, and RL-based Module 2 was assessed.

4.5.1 Test 1: RL Applications to the Inverted Pendulum Example

The inverted pendulum problem is a well-known problem in RL. The goal of the problem is to hold or swing up the pendulum at/to the upright position as depicted in Figure 4.32. The swing-up task is more challenging to learn the control policy.

Table 4.6: Pendulum Properties

Properties	Values
$m\ (kg)$	1
$l\ (m)$	1
$b\ (kg \cdot \frac{m^2}{s})$	0.1

The pendulum model was simulated in Matlab. The pendulum dynamics is shown below:

$$I\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u \tag{4.24}$$

where $I = ml^2$ is the inertia, m is the mass, $g = 9.81\frac{m}{s^2}$ is the gravitational acceleration, l is the pendulum length, b is damping coefficient, and u is the control torque. Table 4.6 shows the pendulum system properties.

Since RL is a model-free optimal control method, the dynamics equation was only used for simulations producing state measurements. The state measurement, then, were used to evaluate rewards at each learning time.

Based on the dynamics, two states, $x = \left[\theta, \dot{\theta}\right]$, and one input, $u \subset [-2, 2]$, were used in the RL formulation. Test cases were run for 2,400 simulation steps per each episode. Simulation step size was set to 0.05 seconds.

Q-learning

Since Q-learning is the most stable algorithm in RL, the swing-up task was tested. The initial condition for the swing-up task was:

$$\begin{aligned} \theta\,(0) &= 0 \\ \dot{\theta}\,(0) &= 0 \end{aligned} \tag{4.25}$$

Continuous states were discretized into 40,401 state features. A pair of two states is one state feature. Continuous action was discretized into 5 actions: $[-2, -1, 0, 1, 2]$.

In this example, the reward was set as shown in Eq. 4.26 and 4.27. Since it was a negative reward, it was a cost. $R(t)$ was composed of quadratic costs based on deviations from the desired state, $[\pi, 0]$. R_{bonus} was a reward when the states were close enough to the desired state.

$$R(t) = -(\theta(t) - \pi)^2 - 0.25 \times \dot{\theta}^2(t) + R_{bonus} \tag{4.26}$$

$$\begin{aligned} &R_{bonus} = 100, \quad \text{if } (\theta - \pi)^2 + \dot{\theta} < 0.01 \\ &R_{bonus} = 0, \quad \text{otherwise} \end{aligned} \tag{4.27}$$

Table 4.7 is a partial result of the Q-value table for the pendulum example. Each row represents different state features: a state combination of θ and $\dot{\theta}$. There are 40,401 rows in total. Columns are for each different action choice. The body of the table is a Q-value for each action. For instance, the fourth action (1) has the best Q-value (17.53) for the first row of the state feature.

The value, not the Q-value, is the best Q-value for each state feature. Figure 4.33 showed the converged value function on a 2-D state space. Different color codes represent values for each state feature; dark blue is low value, and bright yellow is high value. The value function clearly shows that the states around $[\pi, 0]$, which was the goal condition, have the highest value. The initial condition scored a low value as expected.

DQN

Both cases of holding and swinging up to the upright position were tested with DQN. The reward design was the same as the Q-learning test case as Eq. 4.26. Five input choices

Table 4.7: Q-value table (partial)

State Features	Actions				
	-2	-1	0	1	2
	-19.57	-19.08	-18.41	17.53	-19.10
	-11.30	-19.41	-18.43	-17.61	-16.71
	-16.54	-13.63	-16.49	-13.00	-13.96
	-15.94	-14.03	-14.19	-15.04	-12.04
	-13.11	-6.81	-13.18	-14.56	-14.13
	-11.57	-10.33	-10.75	0.60	-10.49
	-9.27	-11.50	-9.89	-9.73	-0.81
	-6.67	44.10	-10.72	-5.51	-5.12
	-5.37	-6.55	4.19	-4.26	-4.36
	0.57	-9.12	-4.00	-4.85	-4.24
	-5.14	19.29	-6.08	-4.27	-5.62
	1.46	-4.88	-5.78	-4.92	-4.99
	0.74	-6.08	-4.75	-4.77	-4.89

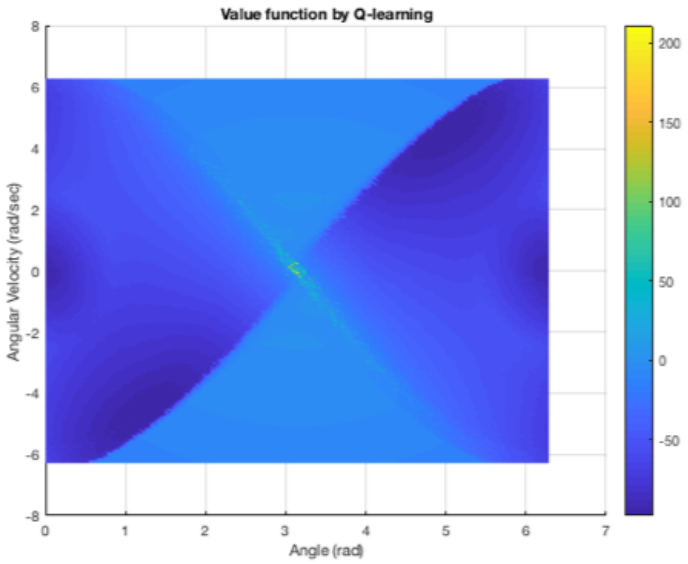

Figure 4.33: Converged value function for swing-up by Q-Learning

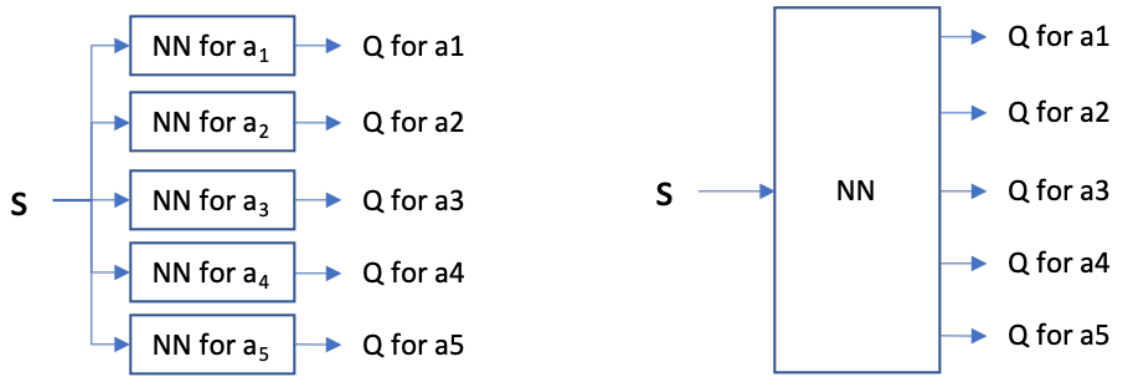

Figure 4.34: Multiple NN (left) vs. Single NN (right)

were set up, either.

The stochastic two-hidden-layer NN was used for the value function approximation: 10 nodes per each hidden layer. In order to optimize NN weights, Adam's method was implemented for the step size. For activation functions, hyper-tangent and linear functions were used. There are two options: one using multiple NN approximation functions for each control action, and the other using a single NN approximation with multiple outputs for control actions. Figure 4.34 illustrates the two options for the value approximation with NN.

Figure 4.35 shows the value function for holding upright position case with multiple NN functions. The color coding looked similar to the Q-learning test case. Figure 4.36 explains the performance of learning, correspondingly. The x-axis is a sequence of training episode. The y-axis is the number of counts that the bonus reward, R_{bonus}, got 100. The higher, the better. As shown in Figure 4.36, DQN was able to learn the optimal control strategy from around the 300th episode and converged after the 360th episode.

Figure 4.37 and 4.38 are test results for the swing-up problem with multiple NN functions. The color coding for the value looked similar to the Q-learning result, but the learning for the control policy was not successful. The general trend seemed to converge, but it kept coming back to fail repeatedly.

From Figure 4.39 to 4.42 are the results where a single NN was used for the value

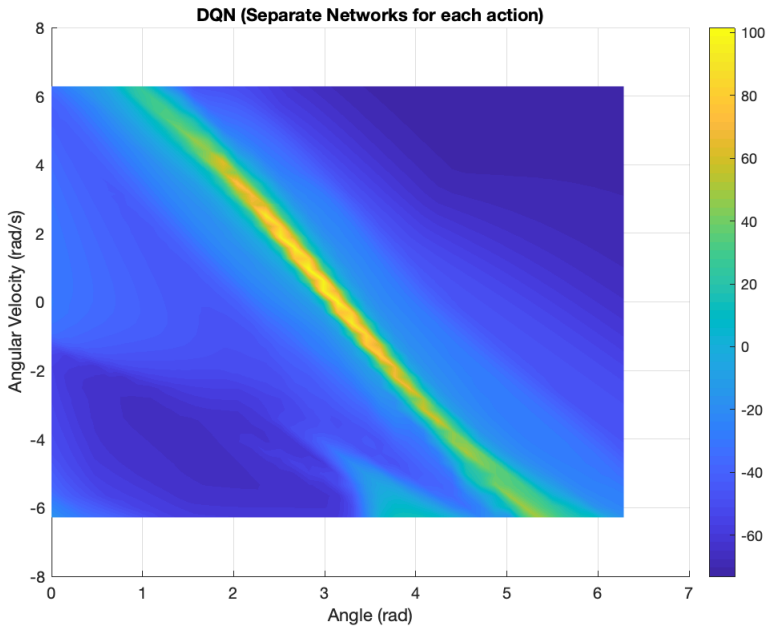

Figure 4.35: Learned value function for holding upright position by DQN (multiple NN)

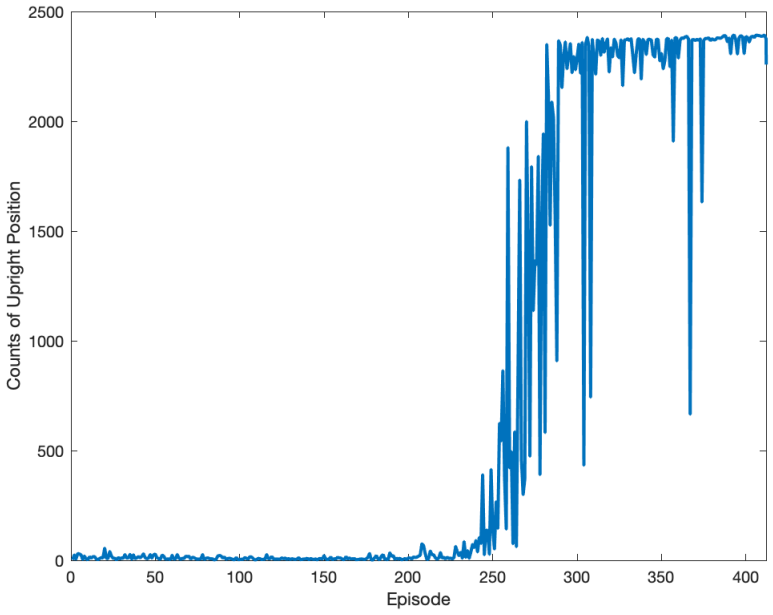

Figure 4.36: Learning performance for holding upright position by DQN (multiple NN)

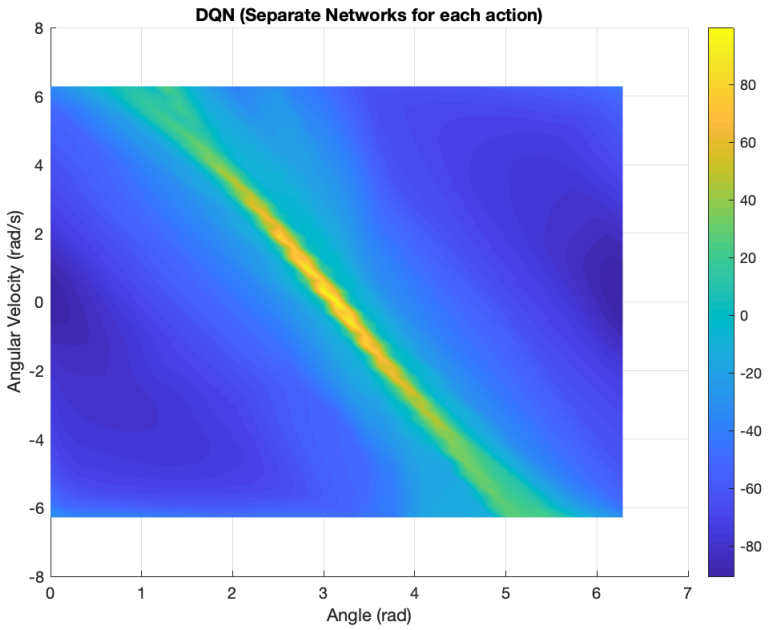

Figure 4.37: Learned value function for swing-up by DQN (multiple NN)

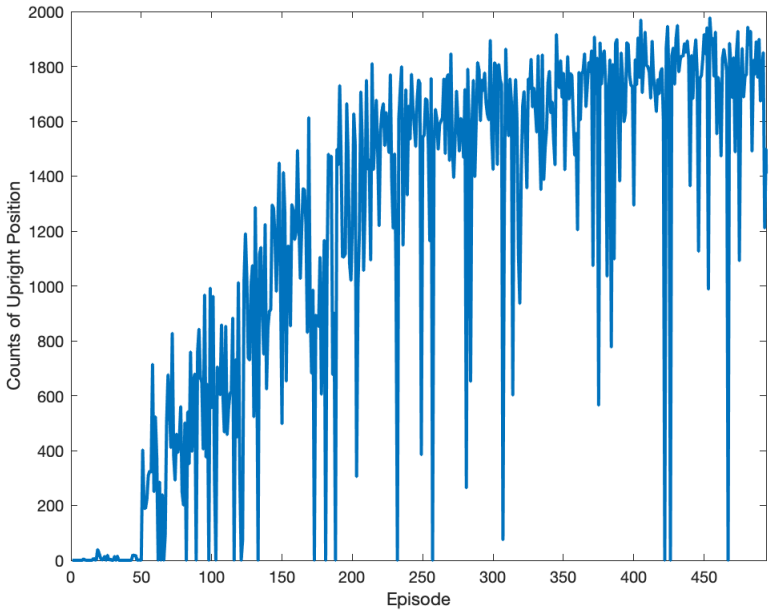

Figure 4.38: Learning performance for swing-up by DQN (multiple NN)

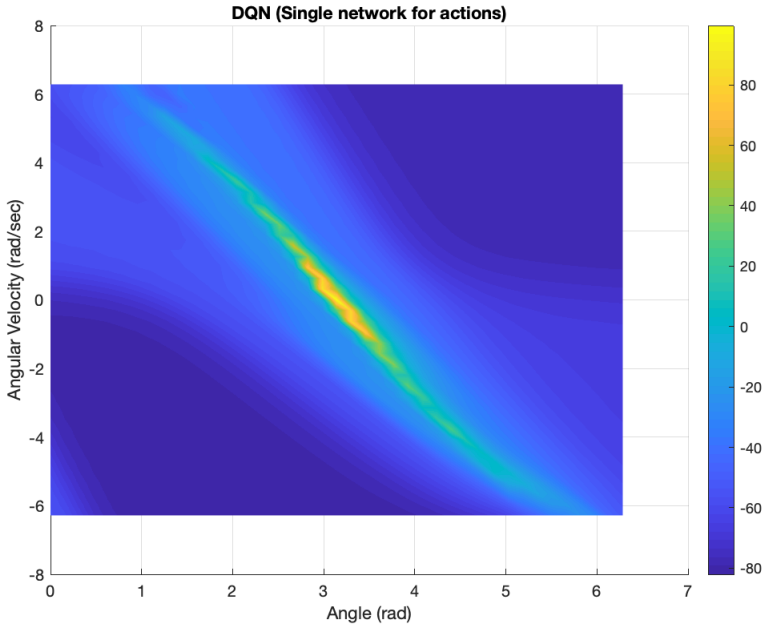

Figure 4.39: Learned value function for holding upright position by DQN (single NN)

function approximation. Just like the multiple NN cases, the control policy for the holding upright position task was successfully learned, but the swing-up test case was not successful.

DDQN

This time, the cost, R, was modified as shown in Eq. 4.28. Only the angular position was the source of the cost. R_{bonus} was set the same as the previous test cases.

$$R(t) = -(\theta(t) - \pi)^2 + R_{bonus} \tag{4.28}$$

With a single NN approximation function, the learning performance was improved significantly. Figure 4.44 shows the successful convergence as well as greater counts of the bonus conditions (over 2,000) than the previous tests.

The test results showed some successful learning behaviors as an optimal control for dynamic systems in test cases. The first three methods were able to find optimal control

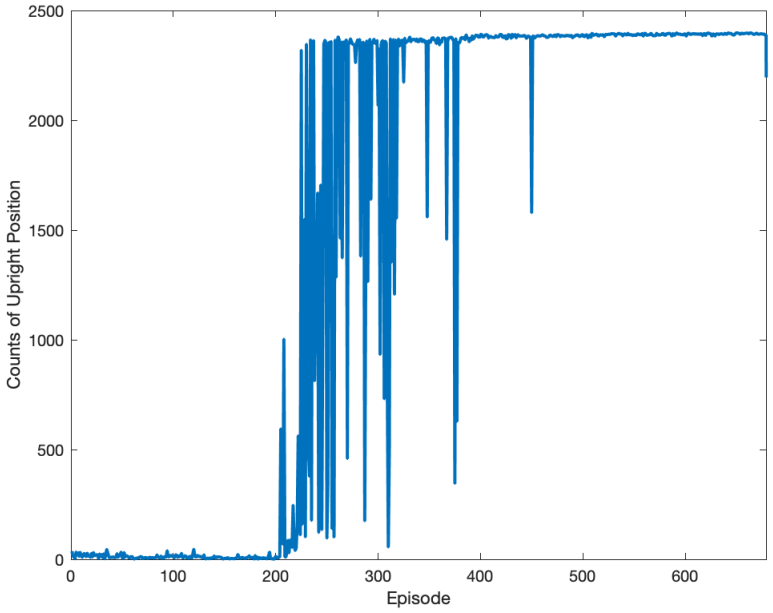

Figure 4.40: Learning performance for holding upright position by DQN (single NN)

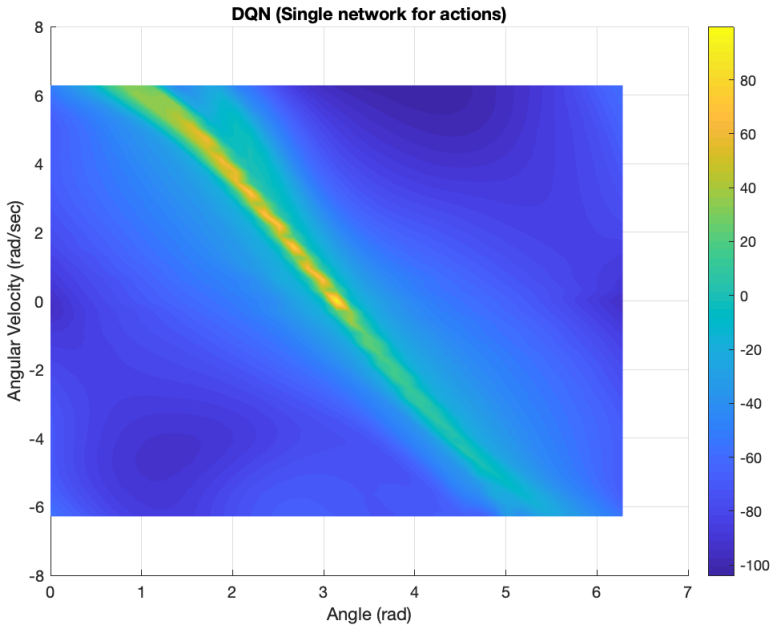

Figure 4.41: Learned value function for swing-up by DQN (single NN)

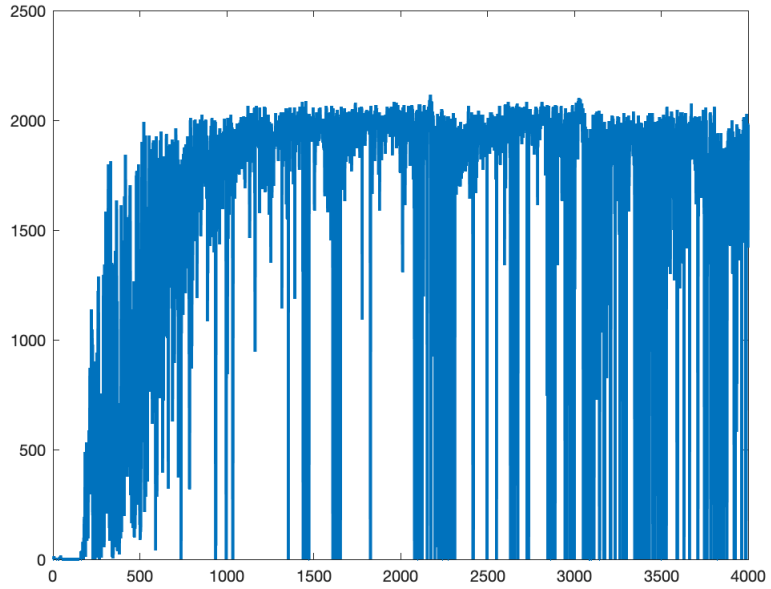

Figure 4.42: Learning performance for swing-up by DQN (single NN)

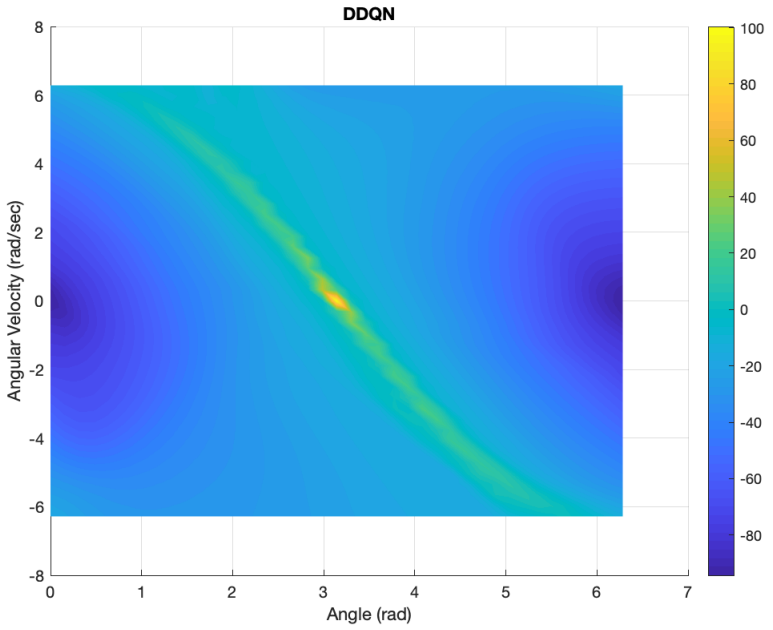

Figure 4.43: Learned value function for swing-up by DDQN (single NN)

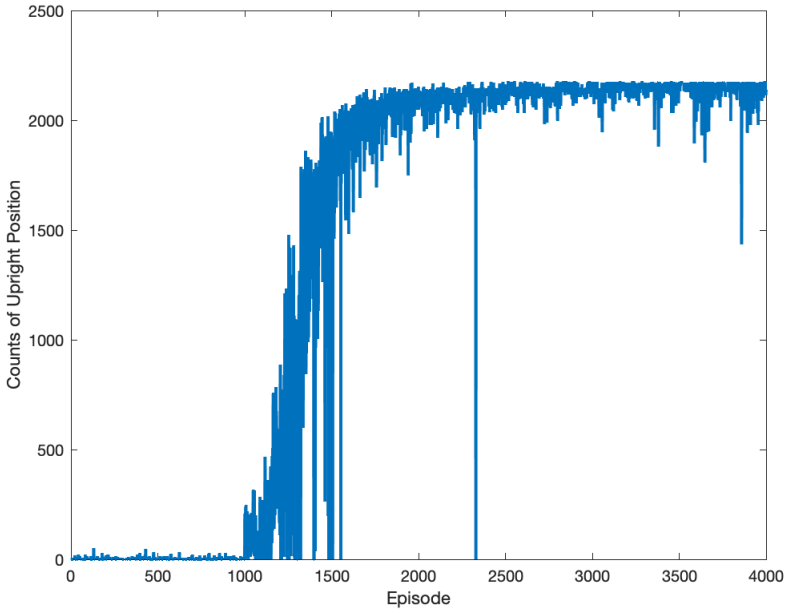

Figure 4.44: Learning performance for swing-up by DDQN (single NN)

paths and corresponding inputs.

However, the test cases also revealed limitations, too. Q-learning and its variants are mainly formulated in discrete state and action spaces. Thus, scalability can be an issue. If a particular problem requires to handle continuous state and action spaces with high dimensionality, it is almost impossible to solve the problem. Also, a control policy is heavily dependent on reward settings in RL, but it is difficult to find proper reward settings for desirable system behaviors because the desirable properties do not explicitly represent the rewards in most cases.

DDPG

The reward, R, was set to be the same as the DDQN experiment. For policy learning, a deterministic gradient ascent method for actor, μ_w, was used.

$$w_{t+1} = w_t + \nabla_w Q_\theta = w_t + \nabla_w \mu_w \cdot \nabla_a Q_\theta \tag{4.29}$$

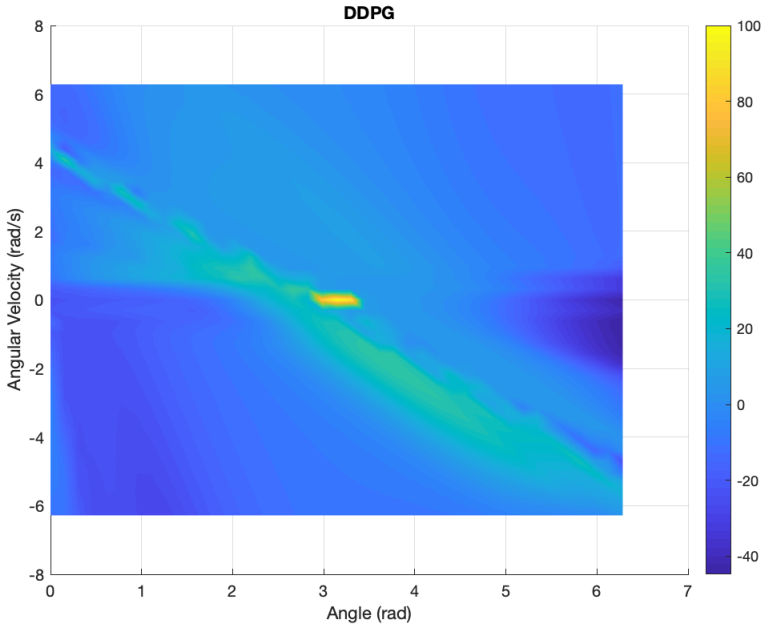

Figure 4.45: Learned value function for holding upright position by DDPG

where ∇_x is a partial derivative for x.

For better convergence behavior, a batch optimization method for critic, Q, was applied.

$$L = \frac{1}{N}\sum_{i=1}^{N}\left(Q^{\text{target}} - Q_\theta\right)^2 \tag{4.30}$$

Figure 4.45 and 4.46 are test results for the holding upright position problem. The value function is now quite different from the ones from other methods, and an optimal control strategy was never learned as shown in Figure 4.46: the number of counts that the pendulum was upright at control time decreased as the training was performed and never went up to the level where DQN was able to reach (near 2400).

It turned out that the NN approximation for control policy went to the extreme of hyperbolic tangent activation function in the output node. It is a well-known problem in NN approximation. There are no apparent causes of this phenomenon, but there are general checkpoints: the selection of activation functions, the number of layers and nodes, and the size of training data. More trials of different combinations of them also did not work,

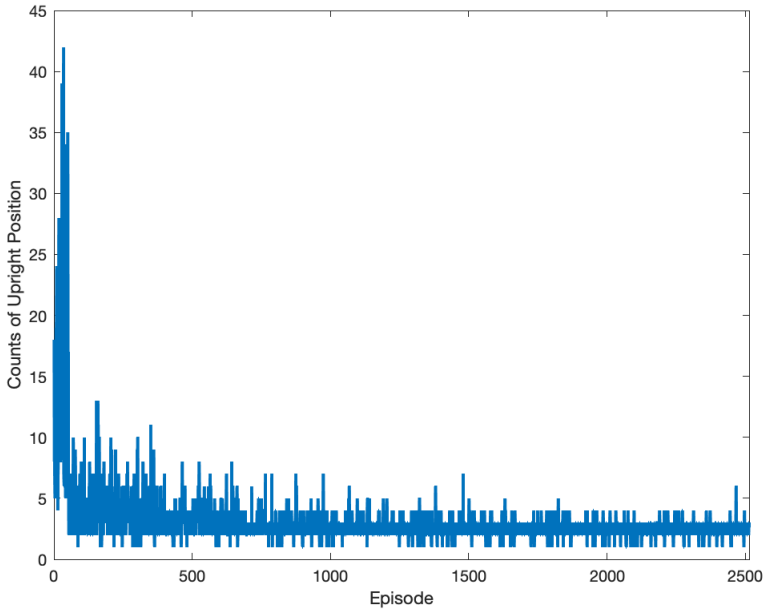

Figure 4.46: Learning performance for holding upright position by DDPG

though.

4.5.2 Test 2: RL applications to the nominal (healthy) hovercraft example

In order to test Hypothesis 2-2, learning behaviors of RL were observed, and the hovercraft waypoint-following test mission evaluated the learned control policy for the adaptation parameter. For simplicity, a single-waypoint mission was introduced. The mission length was set to 50 meters. The desired mission performance for the test scenario was to reach the destination as fast as possible. Since Test 2 runs in nominal hovercraft without fault induced, there will be no failure cases. However, the complete mission time will indicate whether the solution is optimal or not. On top of that, since Experiment 2 showed the impact of the adaptation parameter on mission time, the minimum value of ρ_R is naturally expected as an optimal control strategy.

As stated, Q-learning requires discrete state features and actions. The hovercraft example, however, is in continuous state and action spaces; thus, the Q-learning was not applicable due to its scalability limitation. Suppose that five states are introduced: mission

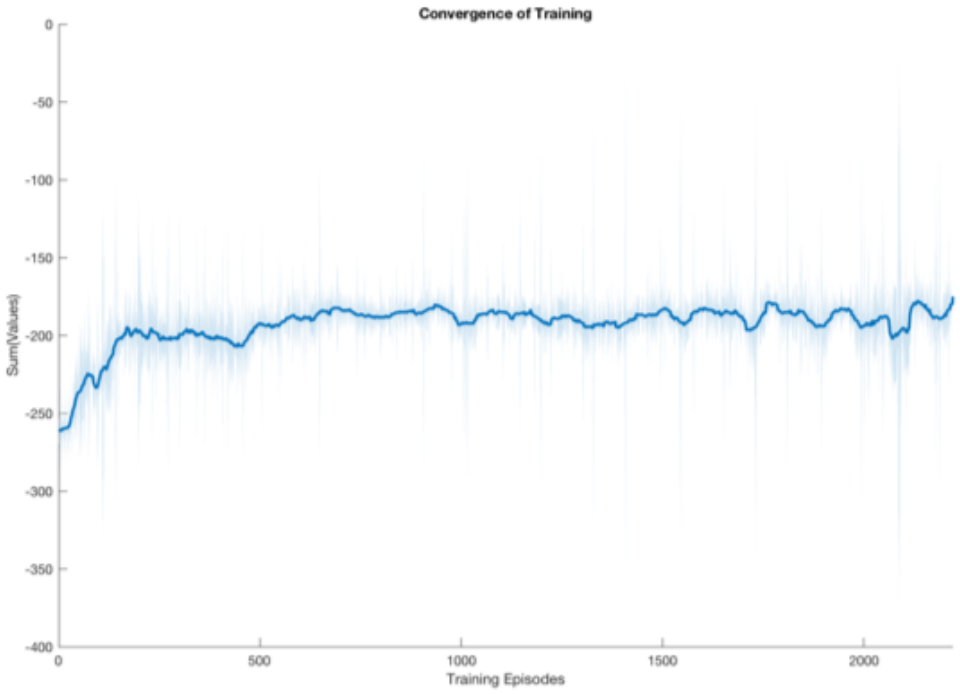

Figure 4.47: Learning performance for the healthy hovercraft example by DQN

length left, angle to the target, x-directional, y-directional, and angular velocity. If they are discretized into 50, 30, 30, 30, and 30 levels for each state, then the dimension of the state feature (combinations of each state) becomes 40,500,500. If an action is discretized into 10 action levels, the size of the Q-table is 40,500,500 $\times$ 10. This huge Q-table is not treatable in RL.

In the DQN approach, on the other hand, there is no limitation in states. The input, on the other hand, needs discretization. Five input dimensions were tried: $\rho_R \in [0.1, 1, 10, 50, 100]$. Figure 4.47 shows the values in each episode. The light blue line is the values at each episode, and the bold blue line is a moving average. The general trend (bold blue line) seemed to be improved and converged, but actual value diverged as the training was being executed. The consistent result is depicted in Figure 4.48. The y-axis of the figure is mission complete time for each episode. It was supposed to decrease and converge over time; rather, it fluctuated over the training episode without converging.

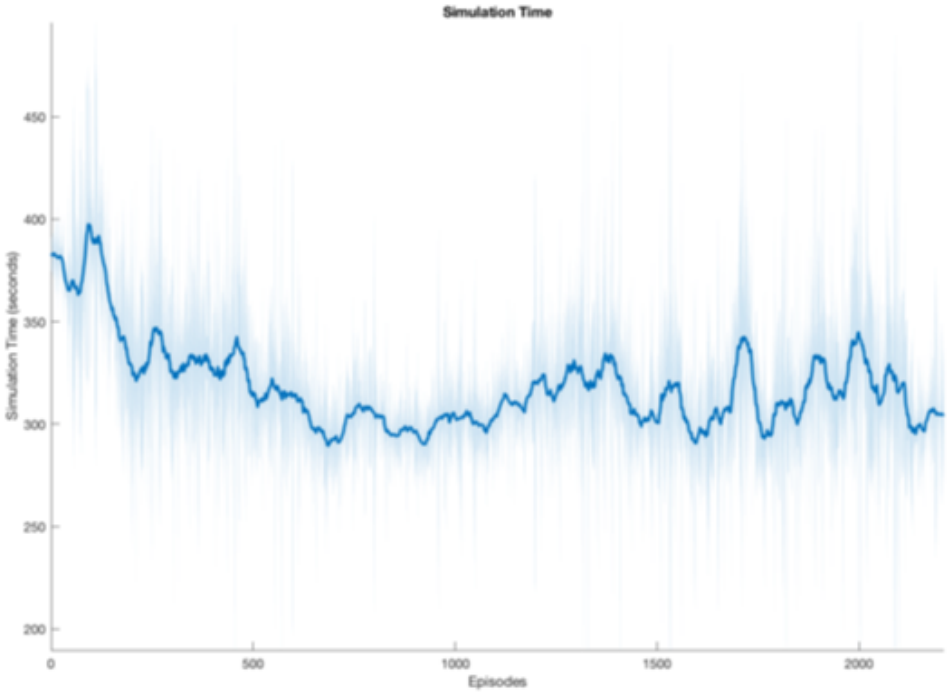

Figure 4.48: Mission complete time for the healthy hovercraft example by DQN

Finally, the DDQN approach was introduced in the healthy hovercraft example. (DDPG was not introduced because DDPG was not even successful in the pendulum example.) Up to the 500^{th} training episode, complete random adaptation parameter values were tried in order to explore more state-action combination space. Figure 4.49 shows the value convergence over the training episode. After the 500^{th} episode, the value jumped up and showed clear converging trend till the end. Figure 4.50 represents the complete mission time in each episode. Considering the setting that the desirable mission capability was minimum mission complete time, Figure 4.50 also shows the consistent result to the converging behavior in Figure 4.49. The complete mission time at episode 4,000 was 245.5 seconds, which is the fastest mission time. Figure 4.50 also proves that DDQN found the optimal solution. It shows the minimum setting of the adaptation parameter, 0.1, throughout the control time steps at the $4,000^{th}$ episode. The DDQN approach was successful in finding an optimal adaptation parameter in the healthy hovercraft example.

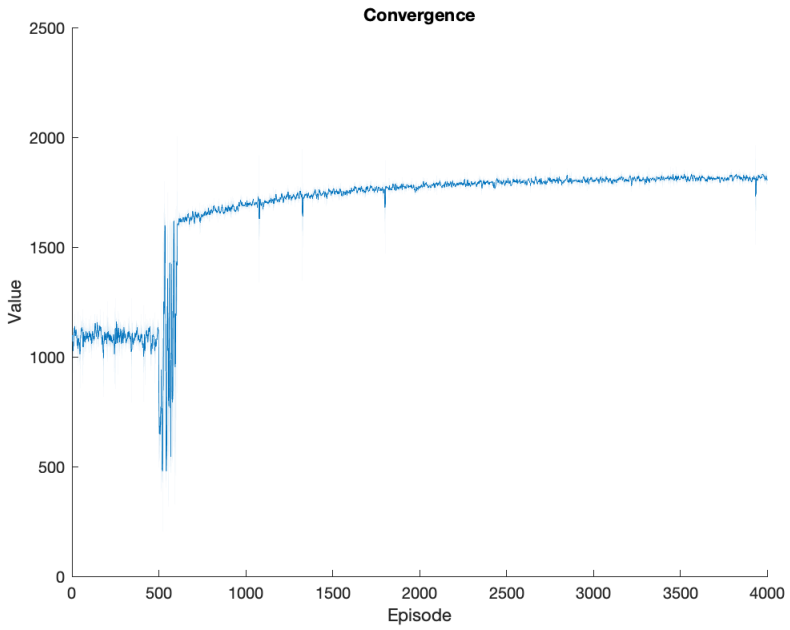

Figure 4.49: Learning performance for the healthy hovercraft example by DDQN

Next test case is to apply the DDQN approach to the faulty hovercraft example. This last test case of Experiment 3 will prove the efficacy of RL to Module 2.

4.5.3 Test 3: RL applications to the faulty hovercraft example

The last test case in Experiment 3 is to apply the DDQN approach to Module 2 in the faulty hovercraft example since the DDQN approach showed a satisfactory learning performance in Test 2.

In Test 3, the hovercraft launched with the right thrust motor fault mode. Initial fault conditions are the same throughout the training episodes. The mission profile was a straight line with a single waypoint as demonstrated in Test 2, but the mission lengths were varied throughout the training episodes to make RL learn the adaptation level for various mission lengths left.

Figure 4.52, unfortunately, represents the unsuccessful learning performance by overlaying successful and failure cases for mission lengths against the training episodes. The blue and red dots are successful and failure cases, respectively. Mission lengths of suc-

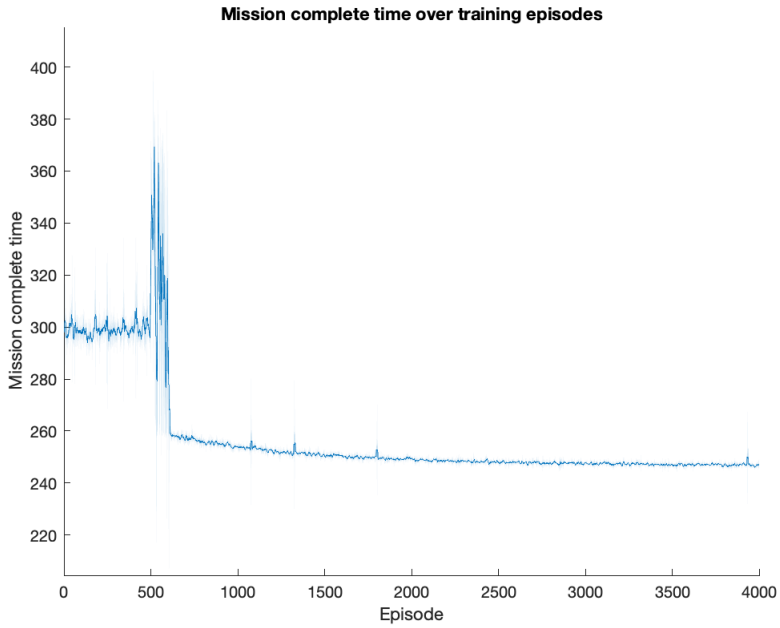

Figure 4.50: Mission complete time for the healthy hovercraft example by DDQN

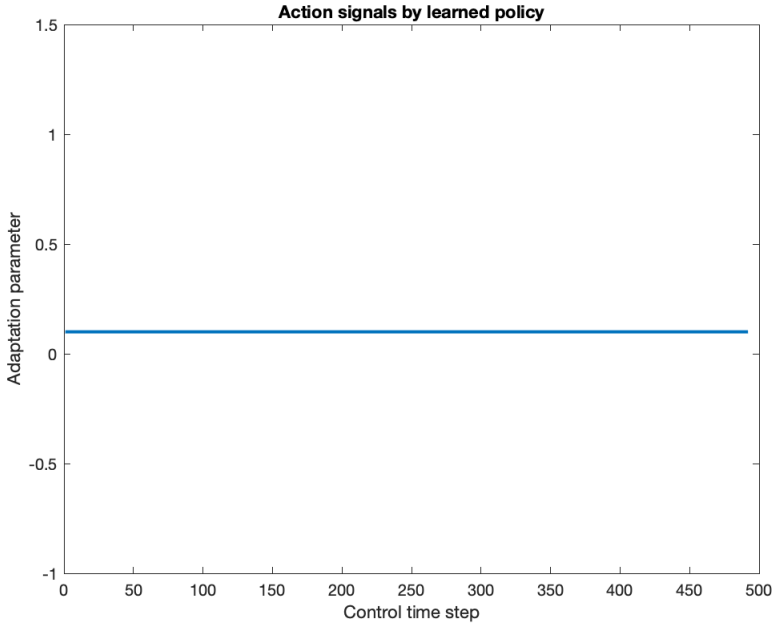

Figure 4.51: Adaptation parameter value sequence using learned policy for the healthy hovercraft example by DDQN

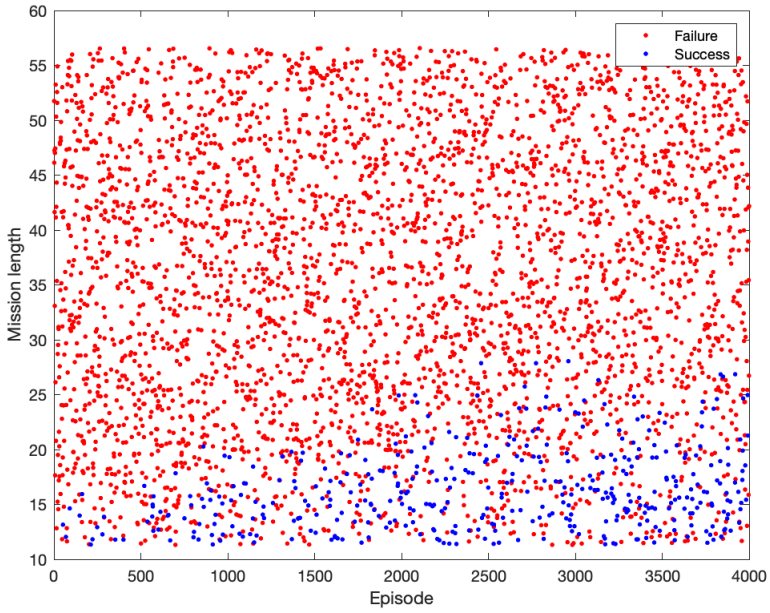

Figure 4.52: Mission success vs. failure over the training episodes for the faulty hovercraft example by DDQN

cessful cases (blue dots) were slightly elongated over the training, but the DDQN module did not learn enough to deal with the fault mode. It only had the fault hovercraft to reach the waypoint for a short range of mission. Moreover, there was no improvement in the probability of success, even for the short mission ranges.

A part of the reasons that the DDQN-based approach in this example was not successful was the failure of competent representation of NN value function approximation. As shown in Figure 4.53, the validation error profile over the training was converged, but it converged into the unacceptable levels.

As expected, Figure 4.54 depicts the failure of a 50-meter traveling distance mission with the hovercraft position trajectory with the DDQN approach. The mission failed at about 20 meters away from the starting point, which is a consistent result to Figure4.52. Figure 4.55 confirms that the fault level grew to the failure limit.

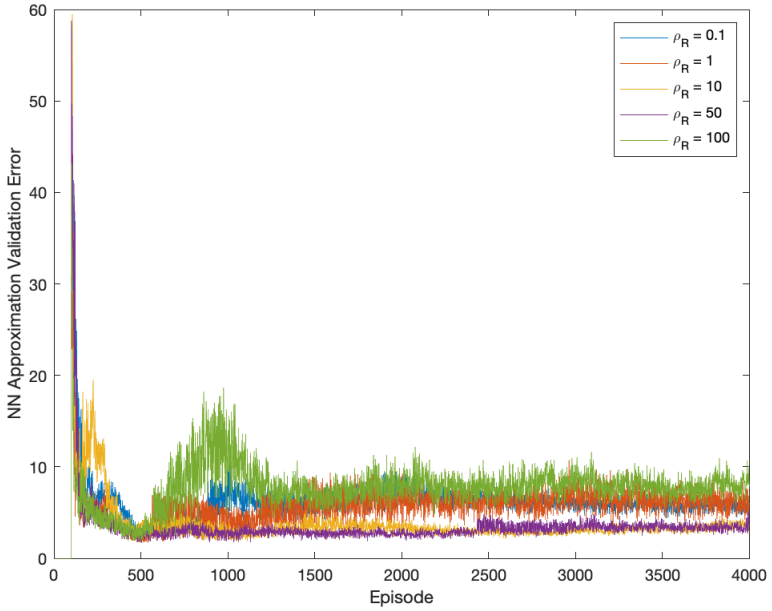

Figure 4.53: NN value approximation validation error over the training episodes for the faulty hovercraft example by DDQN

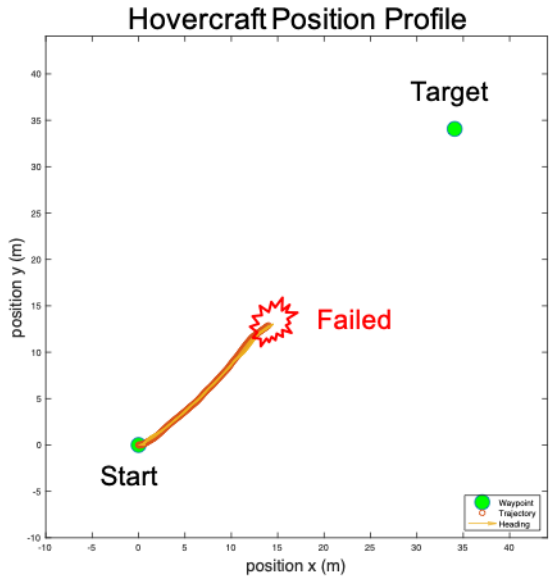

Figure 4.54: Faulty hovercraft trajectory controlled by the DDQN control policy

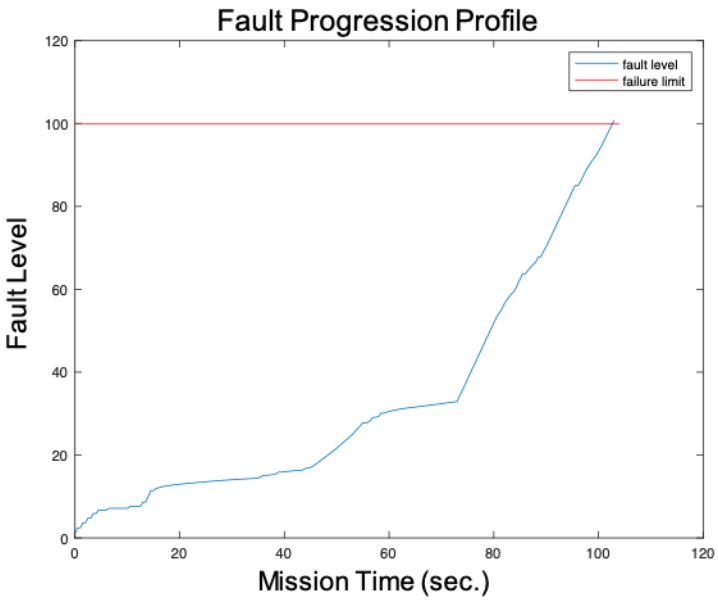

Figure 4.55: Fault level profile over mission time for the faulty hovercraft example by DDQN

4.5.4 Summary of Experiment 3

Results of Experiment 3 failed to show the applicability of RL method for Module 2 in the proposed framework. Experiment 3 was designed to see the effectiveness and potentials of four different popular RL methods - Q-learning, DQN, DDQN, and DDPG - in a simple inverted pendulum example first, and then applied promising methods to the hovercraft example afterward. Even though several test cases showed successful implementations of RL methods to the inverted pendulum and (healthy) hovercraft examples, limitations of RL were observed as summarized below.

- Observations

    1. Q-learning, DQN, and DDQN successfully learned the control policy for the inverted pendulum problem.
    2. There was no noticeable differences between the single NN approximation and multiple NN approximations for input dimensions.
    3. A reward setting was critical to learning performances and control policies.

4. DDPG failed to learn the optimal control policy in the pendulum problem because the NN approximation for policy fell into the extreme.
5. Q-learning was not applicable due to the scalability issue. Discretization of continuous states and actions resulted in an unmanageably huge Q-table.
6. DQN for the healthy hovercraft problem did not show satisfactory results.
7. DDQN for the healthy hovercraft problem did show the successful convergence and optimal control policy for the adaptation parameter.
8. DDQN for the faulty hovercraft problem failed to learn optimal adaptation parameter values due to high validation errors in NN approximation.

4.6 Experiment 4: Simulation-based mission capability prediction modeling and optimization

If Experiment 4 shows that simulation-based long-term mission capability modeling and optimization approach finds an optimal control strategy in terms of ρ_R and make the hovercraft reach the destination in the presence of a critical fault mode, then simulation-based long-term mission capability modeling and optimization approach can be a candidate for Module 2.

According to Experiment 2, the maximum traveling distance available is the most vulnerable long-term mission capability in the presence of the thrust motor fault mode from the hovercraft problem. The vehicle speed is the system performance that is sacrificed by the adaptation parameter.

Figure 4.56 depicts the overall procedure of Experiment 4. First, two essential elements for Module 2 were built: the long-term mission capability prediction model and an optimization module. The mission for the hovercraft was to travel a straight path as long as possible. The mission time was also constrained by 30 minutes. The maximum distances available for states and the adaptation parameter were obtained from the Monte Carlo sim-

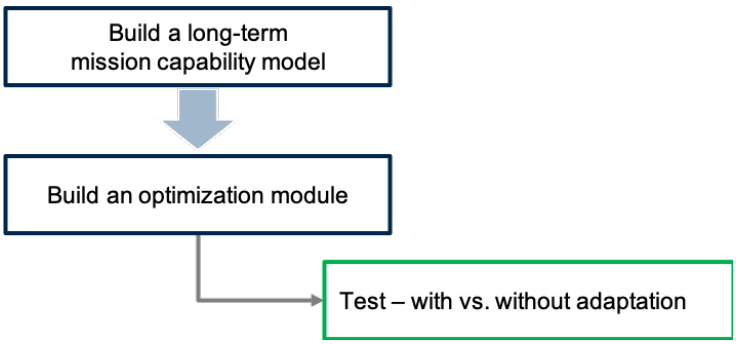

Figure 4.56: Overall procedure and test scenario of Experiment 4

ulations. Then, The maximum distances available were approximated by two-layer NN. This regression function was used in the optimization process during the operations under the presence of the fault mode. Given the observations from Experiment 2, a zero-crossing finding problem was introduced to find a minimum adaptation parameter for the optimization routine because the complete mission time and the adaptation parameter is positively correlated with each other.

Based on Module 2 developed, the test case was set up to examine the effectiveness of the adaptation parameter optimization. The main focus of the test was to compare the probability of mission success between two scenarios: with vs. without adaptation.

For simplicity, the fault growth rate was assumed to be a function of control input. The fault growth model that was used in Experiment 4 is shown below:

$$\dot{x}_f(t) = 0.0025 \times u(t)^2 \tag{4.31}$$

where u is an input, specifically an input voltage, V_{in}, from Eq. 4.2.

Figure 4.57 represents traveling distances available (y-axis) against the states of the fault (x-axis) ranging from 1.26 (healthy) to 100 ohms (failure.) Different color codes are for different adaptation parameter values. In this result, the more significant fault levels, the shorter traveling distances available. Also, the larger the adaptation parameters, the longer traveling distances available in general. However, a plateau area was shown up at

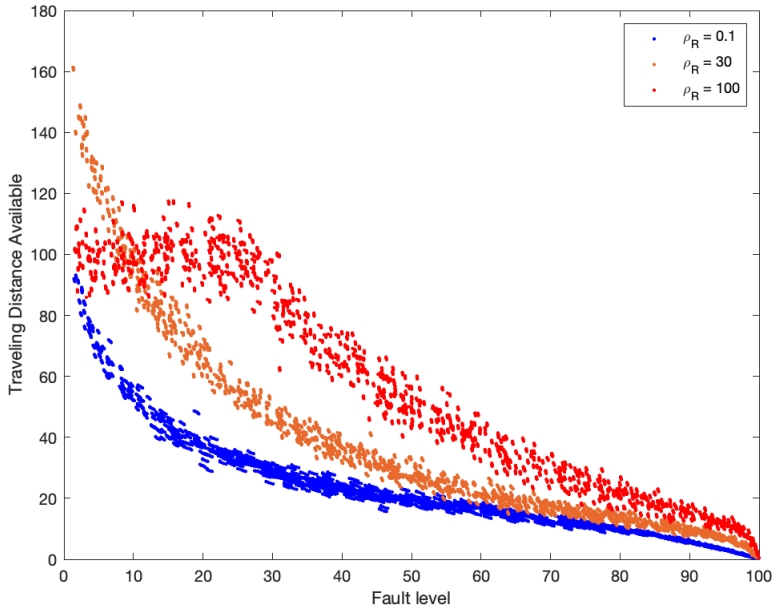

Figure 4.57: Distance available vs. fault level by ρ_R

$\rho_R = 100$. It was because they reached the mission time limit as shown in Figure 4.58. As observed in Experiment 2, Figure 4.58 also confirms that the operation time and the adaptation parameter has a positive correlation in this scenario.

Figure 4.59 shows the histogram of traveling distances available at the fault state, $x_f \in (27.5, 29.5)$ (ohms) by the adaptation parameters. It demonstrates not only the extension of traveling distance available but also increases of variances by higher adaptation parameters. It is closely relevant to the slow vehicle speed and the increase in operating time. The longer the vehicle operates, the more prone to noises the system performances get.

With a Gaussian distribution assumption, the average and standard deviation of the maximum traveling distance available were regressed by NN for x_f and ρ_R. The regression was done in SAS JMP Pro 14.1.0. (In this specific example, the traveling distance available was not dependent on system states, x.) Two hidden-layer NN models were used. The first layer close to the input nodes had six activation nodes: three linear and three hyperbolic tangent functions. The second layer close to the output layer used three linear activation functions. Figure 4.60 shows the quality of the regression. The average prediction was

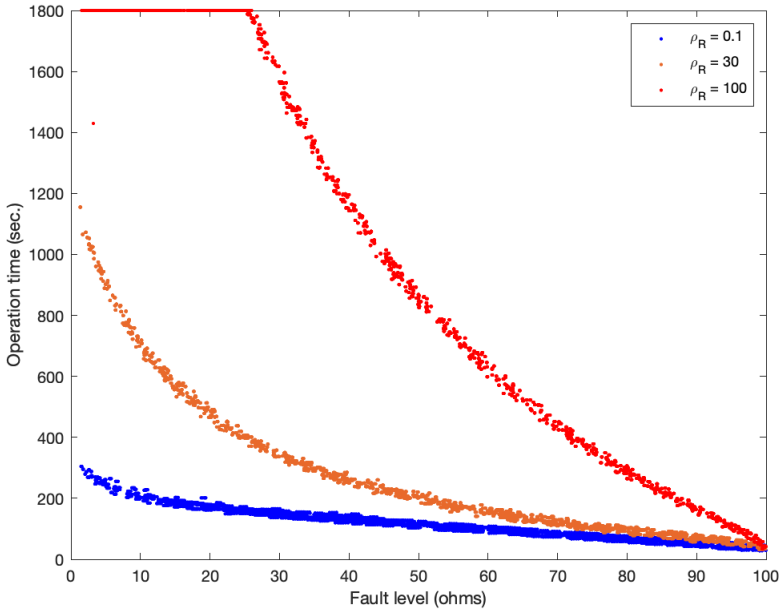

Figure 4.58: Operation time vs. fault level by ρ_R

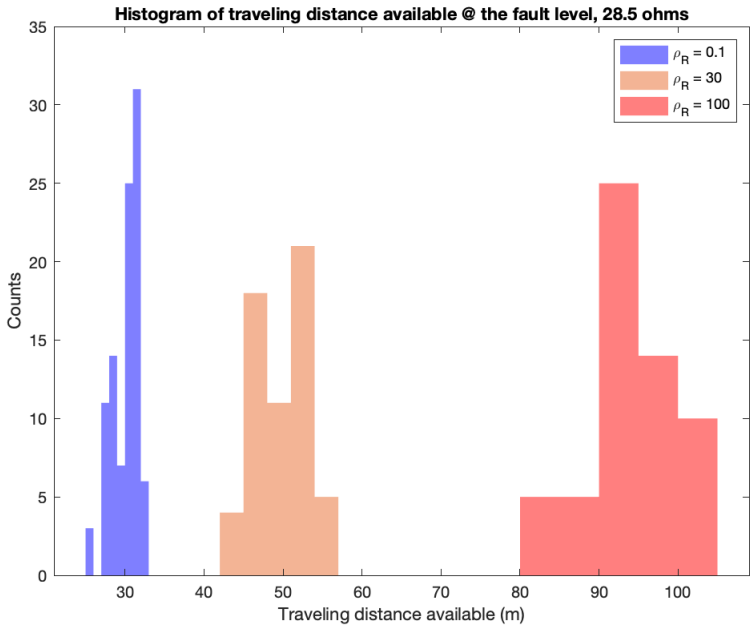

Figure 4.59: Histogram of traveling distance available @ the fault level, 28.5 $\pm$ 1 ohms

able to be regressed accurately, whereas the standard deviation was not as accurate as the average model.

The regression models were drawn on top of their histograms in Figure 4.61. The regressions may not be perfect but seemed good enough to represent the traveling distance available considering system noises.

For the optimization module, a zero-crossing problem was solved since the correlation between the adaptation parameter and the operating time was identified. Equation 4.32 represents the problem to be solved in the optimization step in Module 2 for this example. For the zero-crossing problem, Brent's method was applied. Brent's method is a numerical root-finding algorithm using bisection, secant, and inverse quadratic interpolation methods [103].

$$\text{dist.}_{\alpha}\left(x_f, \rho_R\right) = \text{dist.}_{\text{req.}} + \text{buffer} \tag{4.32}$$

where

$$\text{dist.}_{\alpha} = \mu_{\text{dist.}_{\text{avail.}}}\left(x_f, \rho_R\right) - z_{\alpha} \cdot \sigma_{\text{dist.}_{\text{avail.}}} \tag{4.33}$$

$\mu_{\text{dist.}_{\text{avail.}}}$ is a traveling distance available, α is a confidence level, z_{α} is a standard-z value for α, $\text{dist.}_{\text{req.}}$ is a mission distance left, and buffer is a buffer distance for reserve. The buffer was set to 3 meters.

4.6.1 Test Case: With adaptation vs. no adaptation

In order to compare two cases with or without adaptation capability, a test scenario designed. A mission profile is a straight path with a single waypoint, which is 200 meters apart from the starting point. The mission is complete when the hovercraft reached within a 3-meter range around the waypoint coordinate. A thrust fault mode happens at 70 meters away from the starting point. A default value of ρ_R is 0.1, which is the setting for the hovercraft to move fastest. It was assumed that a fault state is perfectly known.

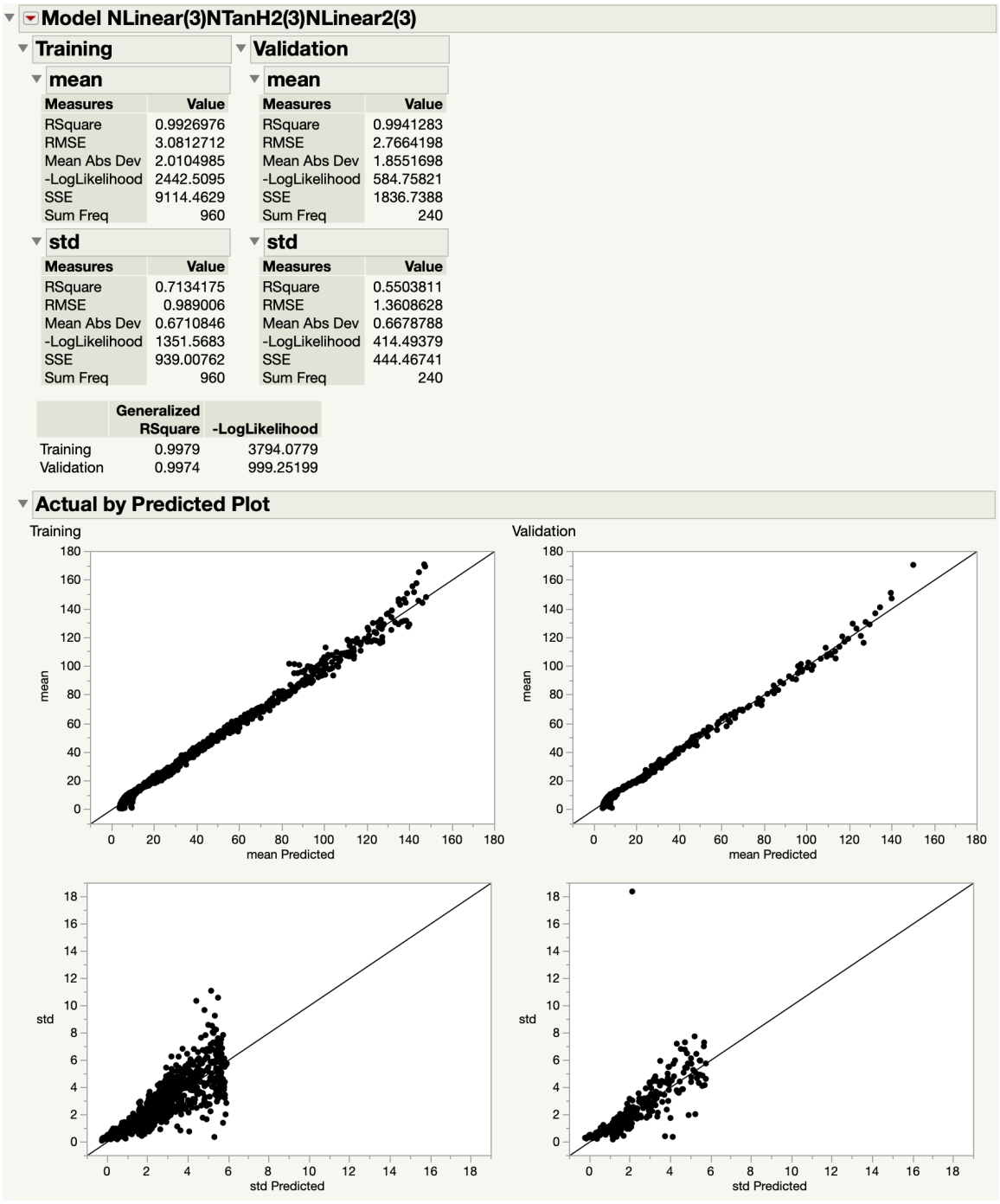

Figure 4.60: NN regression on traveling distance available with respect to x_f and ρ_R

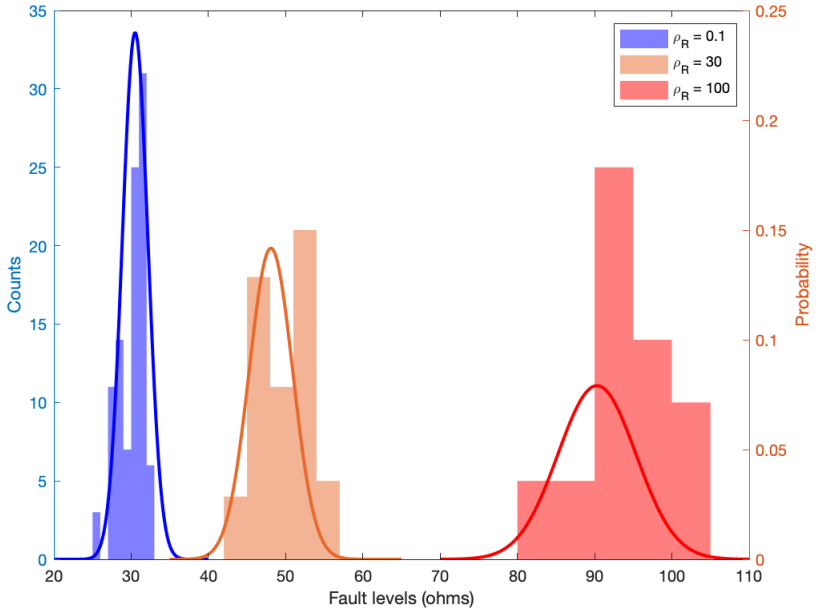

Figure 4.61: Gaussian distribution models of traveling distance available by adaptation parameters laid over histograms

Figure 4.62 shows the effects of the adaptation on system performances and long-term mission capability. The top plot is the predicted distributions of the distance available at the onset of fault. The solid orange curve is the distribution of the predicted distance available if ρR was kept at 0.1, and the orange dashed vertical line is the actual position where the system failed. On the other hand, the solid blue curve is the prediction of the distance available when the adaptation was active (when Brent's method solved the zero-crossing problem.) The solid red vertical line is the destination position.

The second plot is the fault level progression profiles over time. The orange line is the case without adaptation, and it reached the failure limit at around 400 seconds. The blue line is the case with adaptation, and it shows that the fault level did not hit the failure limit when the hovercraft reached the destination.

The third plot is the corresponding adaptation parameter profiles. The orange line was set to constant at 0.1 without adaptation, and the blue line varied over time by the adaptation module. The adaptation profile shows a relatively smooth decrease up to 650 seconds. It

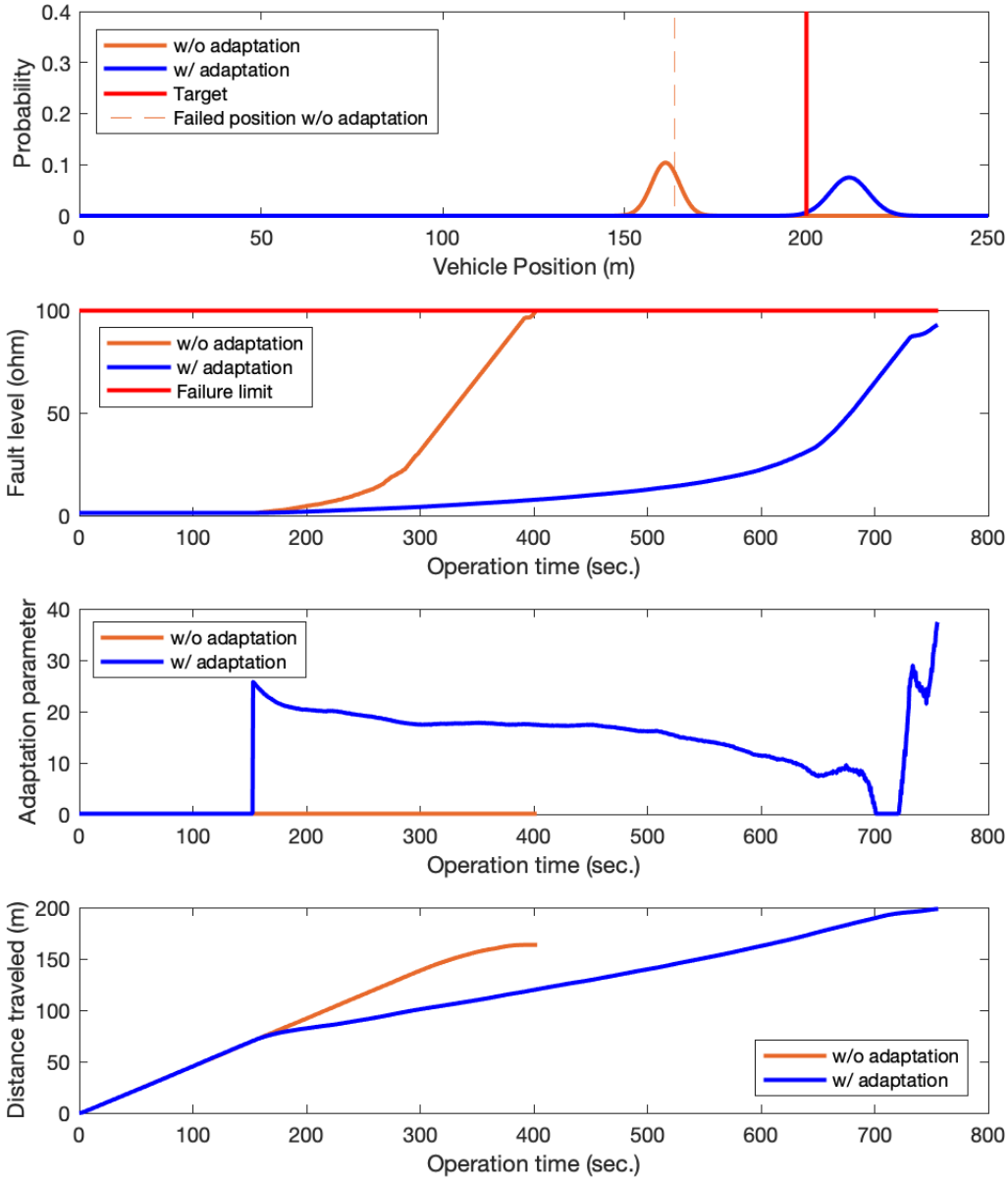

Figure 4.62: With adaptation vs. no adaptation

implies a conservative adaptation at first due to a high prediction variance at the beginning. After 650 seconds, the adaptation parameter varied aggressively because the controller drove hovercraft harder as it got close to the destination range.

The bottom plot shows the distance traveled on y-axis over time. As it is displayed, the vehicle speed (slope of the curve) with adaptation decreased significantly, whereas there was no difference in speed without adaptation. This again implies that the immediate recovery module (Module 1) in this framework works as AFTCS in that the operating point comes back to the design operating point under the presence of a fault mode, but the long-term mission capability module (Module 2) determines a new operating point to extend the vulnerable long-term mission capability.

4.6.2 Summary of Experiment 4

Results of Experiment 4 succeeded in showing the online adaptation of ρ_R by simulation-based mission capability modeling and optimization approach within the proposed framework. Experiment 4 tested the proposed simulation-based mission capability modeling and optimization method for long-term mission capability recovery by the hovercraft waypoint-following problem. Experiment 4 covered the modeling and optimization procedure and compared the case scenarios with and without the long-term mission capability adaptation. The results clearly show the effectiveness of the proposed method in the online long-term mission capability extension and the improvement of the probability of success under the presence of a fault mode.

- Observations

    1. A statistical model for long-term mission capability provided robust prediction concerning uncertainty.
    2. At the early adaptation, a conservative level of adaptation was observed.

3. A long-term mission capability adaptation effectively extended the vulnerable mission capability.
4. Module 2 with the proposed simulation-based long-term mission capability modeling and optimization routine determined a new desirable operating point in the presence of a fault, whereas Module 1 recovered the degraded operating point to the desired operating point.
5. System performance was sacrificed as a new operating point was determined.

4.7 Experiment 5: Hypothesis test for Module 3 (Particle filtering-based fault diagnosis)

Up to Experiment 4, it was assumed that the fault detection and the state of fault were perfectly done and known. In reality, there is no such scenario. Therefore, Module 3 includes a particle filtering-based fault diagnosis approach to detect a fault mode and estimate fault levels correctly. As reviewed in Chapter 2, the particle filtering-based fault diagnosis approach uses both a fault growth model and measurement data; thus, an incorrect fault growth model may degrade the overall quality of fault diagnosis. By recalling Research Question 3-1 and Hypothesis 3-1,

- **Research Question 3-1:** How to improve the performance of the particle filtering-based fault diagnosis routine when a fault growth model is unknown?
- **Hypothesis 3-1:** If online fault growth model estimation is introduced, performance of the particle filtering-based fault diagnosis will be improved when a fault growth pattern is unknown.

Experiment 5 was designed to test whether the detection of a fault mode would be affected by the accuracy of the fault growth model. If Experiment 5 shows that online fault growth model estimation improves detection time and fault state estimation by using a particle filtering-based fault diagnosis, then it will support Hypothesis 3-1.

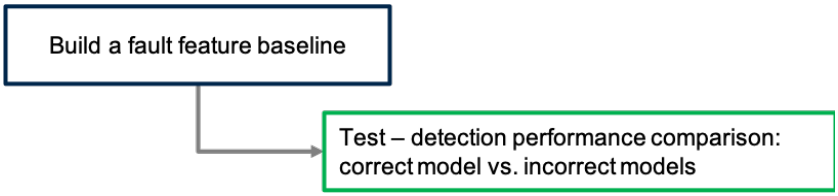

Figure 4.63: Overall procedure and test scenario of Experiment 5

An electrical motor described in Section 4.1 in this chapter was used as a test application. This motor model is the same motor model that was used in simulations of Experiment 3 and 4 and will be used in Experiment 6 and 7, too. The fault feature was obtained by Eq. 4.12. Measurement noises were assumed to follow Gaussian distributions as shown in Eq. 4.34. Figure 4.63 describes the overview of Experiment 5.

$$R_m(t) = \frac{V_{in}(t) - k_e \Omega(t)}{\Omega^2(t) \cdot \frac{b}{k_t}} \qquad \text{(4.12 revisited)}$$

$$\begin{aligned} \Omega &\sim N(0,1) \\ V_{in} &\sim N(0, 0.03) \end{aligned} \qquad (4.34)$$

Figure 4.64 shows a baseline fault feature (bottom) along with the control input voltage (top), corresponding motor shaft angular velocity (middle) profiles. According to Eq. 4.12, Ω is in the denominator. It means that if there is no or very weak input signals, V_{in}, then the fault feature will diverge. The red circles in Figure 4.64 clearly show this phenomenon. It is natural because a fault can be seen only when the faulty system is running unless the system is stripped down for maintenance.

In this regard, the variance of fault feature varies against the control input, V_{in}. Figure 4.65 and Figure 4.66 show the feature data against V_{in} and fault feature in pdf, respectively. In the particle filtering-based fault diagnosis module, therefore, used different baseline statistics based on V_{in}. The fault feature was not estimated if $V_{in} < 1$.

Having the baseline statistics, an insulation degradation fault mode was induced at 600

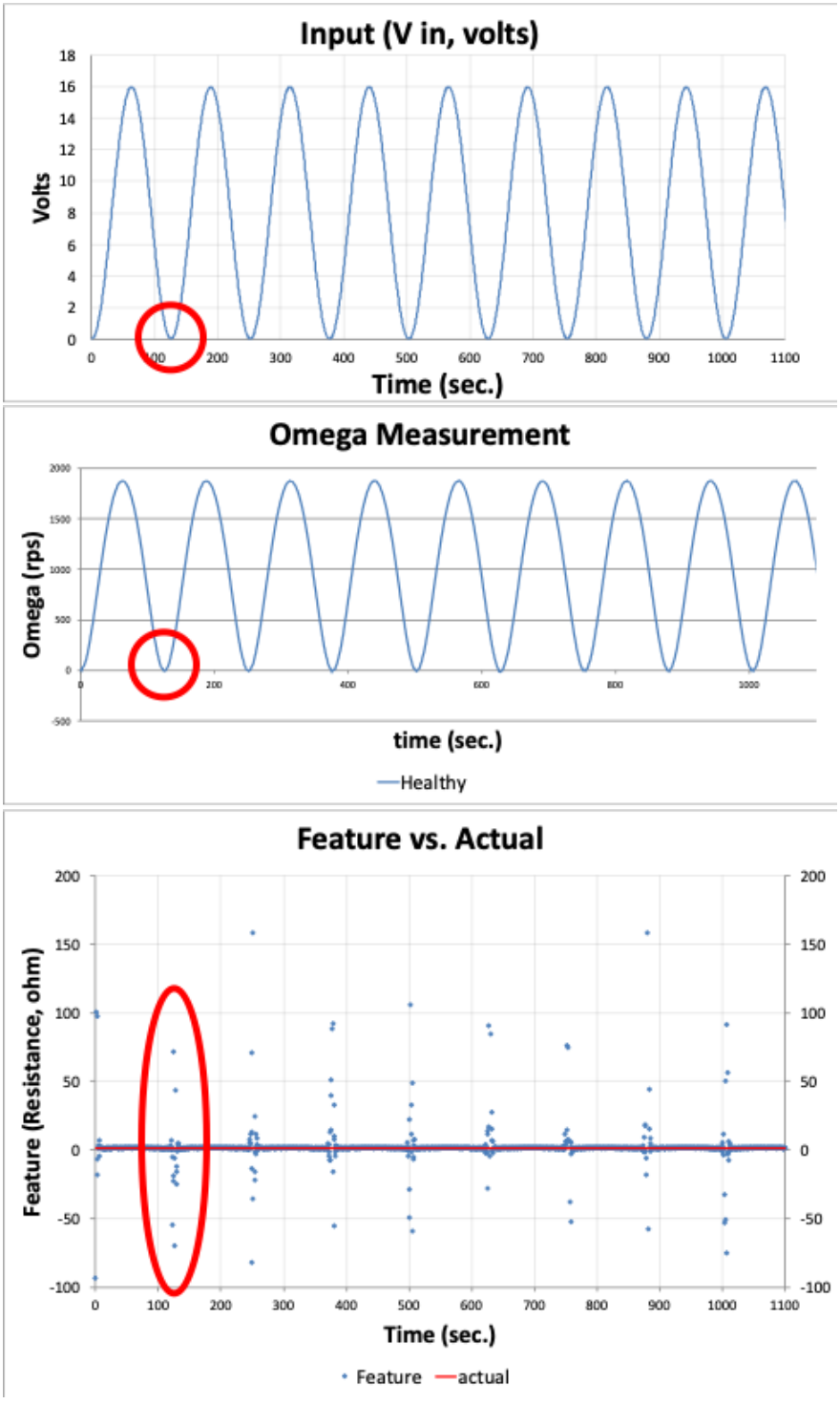

Figure 4.64: Input signal, measurement, and fault feature for baseline

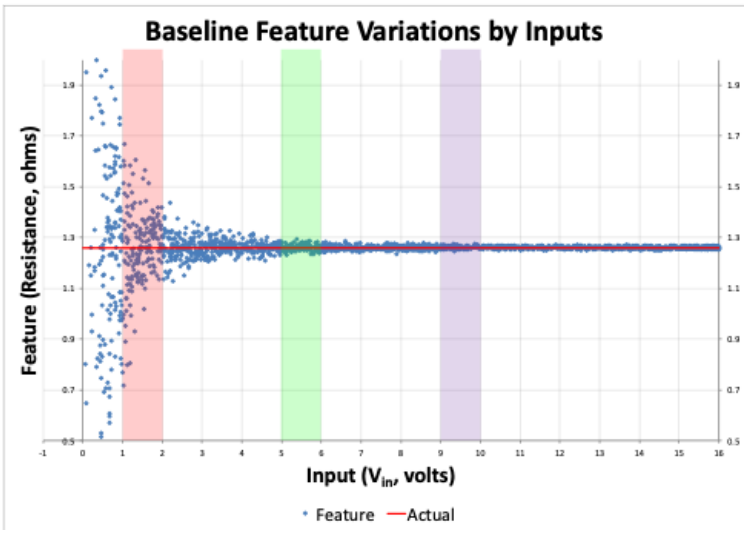

Figure 4.65: Fault feature baseline variances with respect to V_{in}

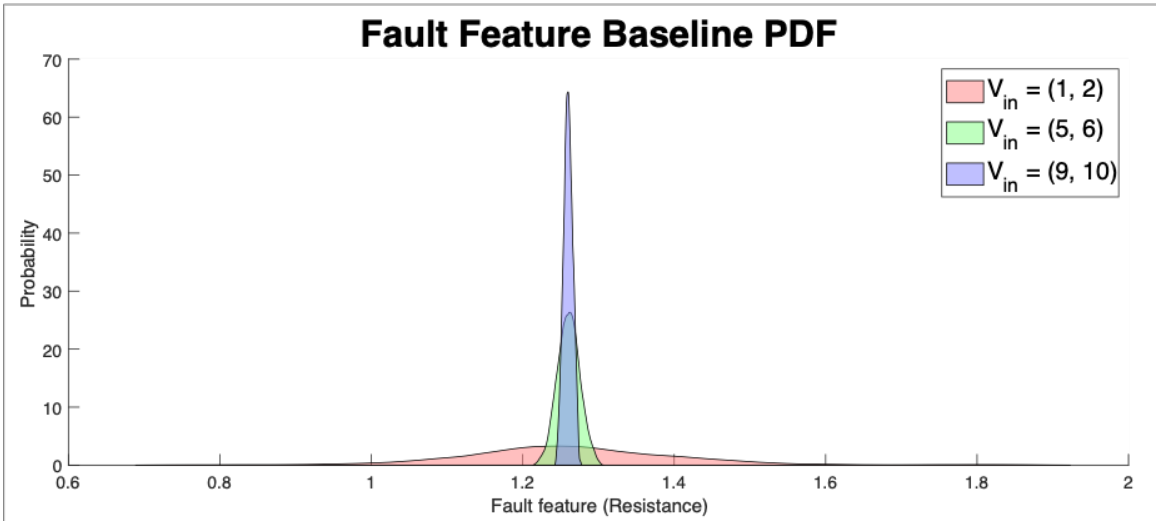

Figure 4.66: Fault feature baseline pdf by three different ranges of V_{in}

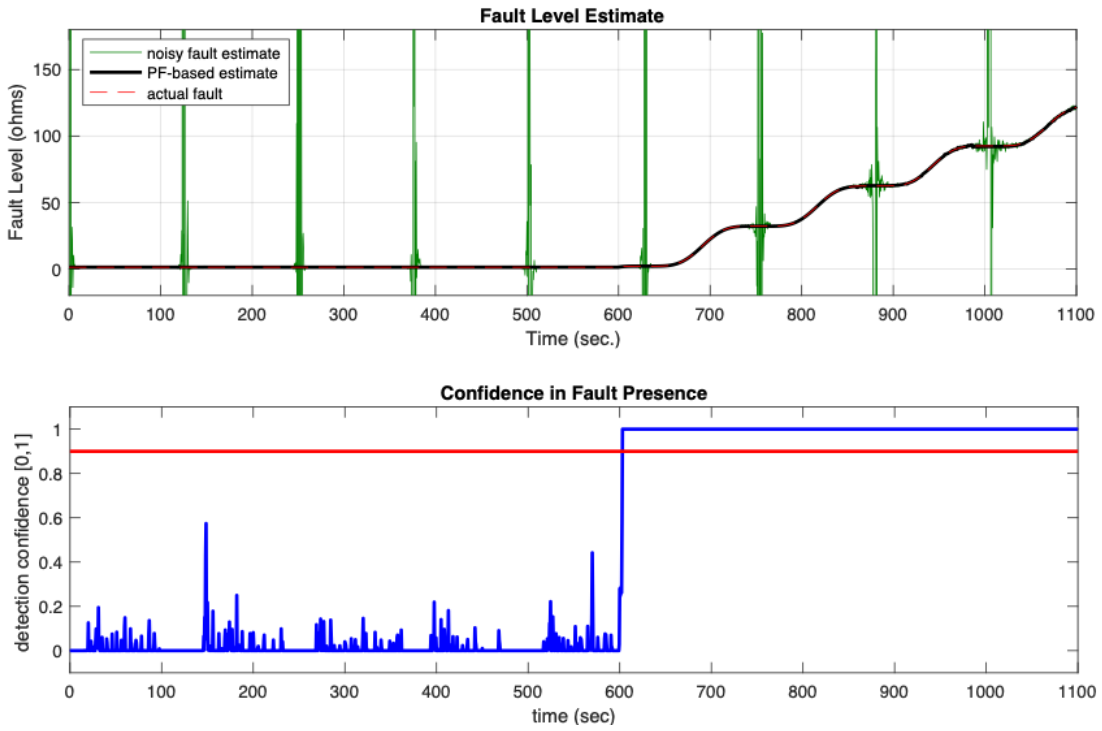

Figure 4.67: Particle filtering-based fault diagnosis result with the accurate fault growth model

seconds. The simulated fault growth model was the same model that was used in Experiment 4:

$$\dot{x}_f(t) = 0.0025 \times u(t)^2 \qquad \text{(4.31 revisited)}$$

x_f is the right motor resistance, R_m, and u is the control input voltage, V_{in}.

Figure 4.67 is the results of the particle filtering-based fault diagnosis routine. The top plot shows the raw fault feature (green), estimate (black), and the actual fault level over time. The estimate was not bothered by the noisy raw signals. The bottom plot represents the confidence of the fault presence (blue line.) The red straight line is the 90 percent confidence level, which is the fault detection threshold in this example. In this test case, the fault was detected at 603 seconds.

Figure 4.68 shows pdfs of baseline and estimate before (the left plot recorded at 525 seconds) and after (the right recorded as soon as the fault was detected) fault. Before

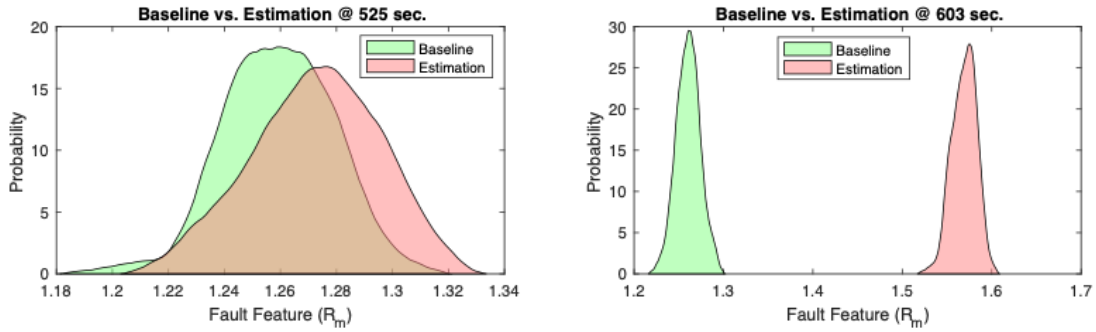

Figure 4.68: Baseline vs. estimate pdfs before detection (left) and right after detection (right)

the fault occurred, the majority of the fault estimate pdf overlapped with the baseline pdf, which implies the condition at 525 seconds was not distinguished from the healthy state. At 603 seconds, the estimate pdf was away from the baseline, indicating that statistically 100 percent certain that the condition deviated from the healthy condition.

4.7.1 Test: Correct model vs. incorrect models

Incorrect fault growth models were introduced in the same particle filtering-based fault diagnosis routine. The simulated accurate growth model is Eq. 4.31 again. The incorrect models are:

1. Model 1: $\dot{R}_m(t) = 0.0125 \cdot V_{in}^2(t)$
2. Model 2: $\dot{R}_m(t) = 0.0005 \cdot V_{in}^2(t)$
3. Model 3: $\dot{R}_m(t) = V_{in}^2(t)$
4. Model 4: $\dot{R}_m(t) = 0.02 \cdot V_{in}(t)$
5. Model 5: $\dot{R}_m(t) = 0.1$

Fault growth model 1 and 3 are faster growth rates than the simulated model. Model 2 is slower. Model 4 and 5, on the other hand, had even different fault growth patterns.

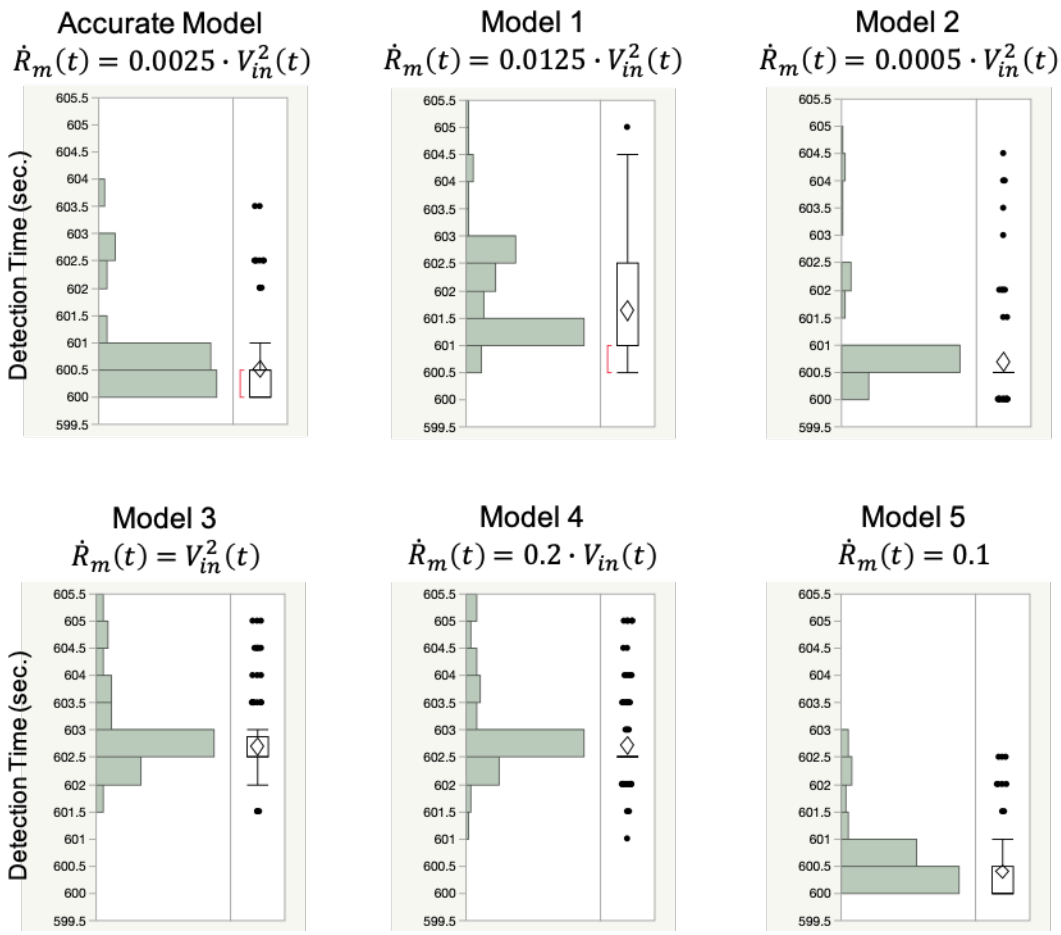

Figure 4.69: Fault detection time comparisons with respect to fault growth model accuracy

Figure 4.69 shows histograms of fault detection time. Figure 4.70 is cdfs of fault detection time. Even though the models were quite different from the correct one in the growth rates and patterns, the fault detection times for each model were not that different from the detection time by the correct model.

Although the fault detection time was not significantly affected by the accuracy of the fault growth model, fault state estimation could be substantially impaired. Figure 4.71 shows the fault estimation profiles over time when each incorrect fault growth model was used in the particle filtering-based diagnosis routine. Model 1 and 4 showed slight deviations from the actual fault levels. Model 3 caused significant errors for certain period times. It led the conclusion that online fault growth dynamics identification is necessary after all.

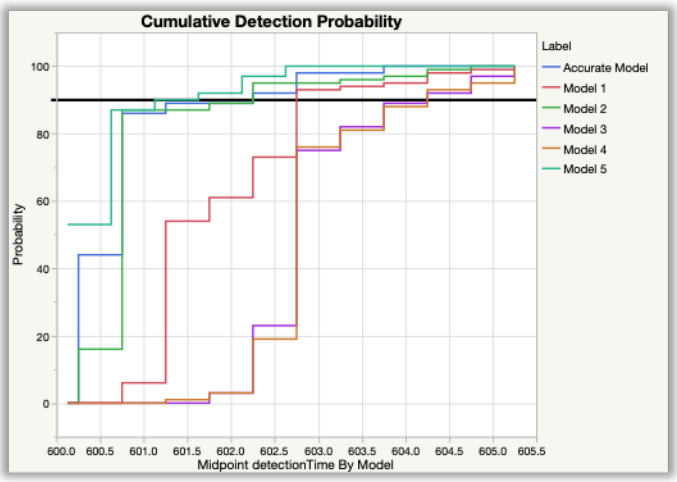

Figure 4.70: Fault detection time comparisons with respect to fault growth model accuracy (CDF)

4.7.2 Summary of Experiment 5

Results of Experiment 5 supported Hypothesis 3-1 by showing that online fault growth model estimation improves the performance of a particle filtering-based fault diagnosis approach. Various fault growth models were introduced in the particle filtering-based fault diagnosis routine and compared the fault detection and estimation performances with each other. Although the fault detection time was not significantly affected by the inaccurate models, there seemed to be a chance that the fault level estimation could be significantly impaired. The observations from Experiment 5 were summarized below:

- Observations

    1. A particle filtering-based fault diagnosis routine provided a statistical measure of fault detection and fault level estimation.
    2. A fault growth model did not significantly delay the fault detection time.
    3. The test showed that the accuracy of the fault growth model did affect the quality of fault level estimation.

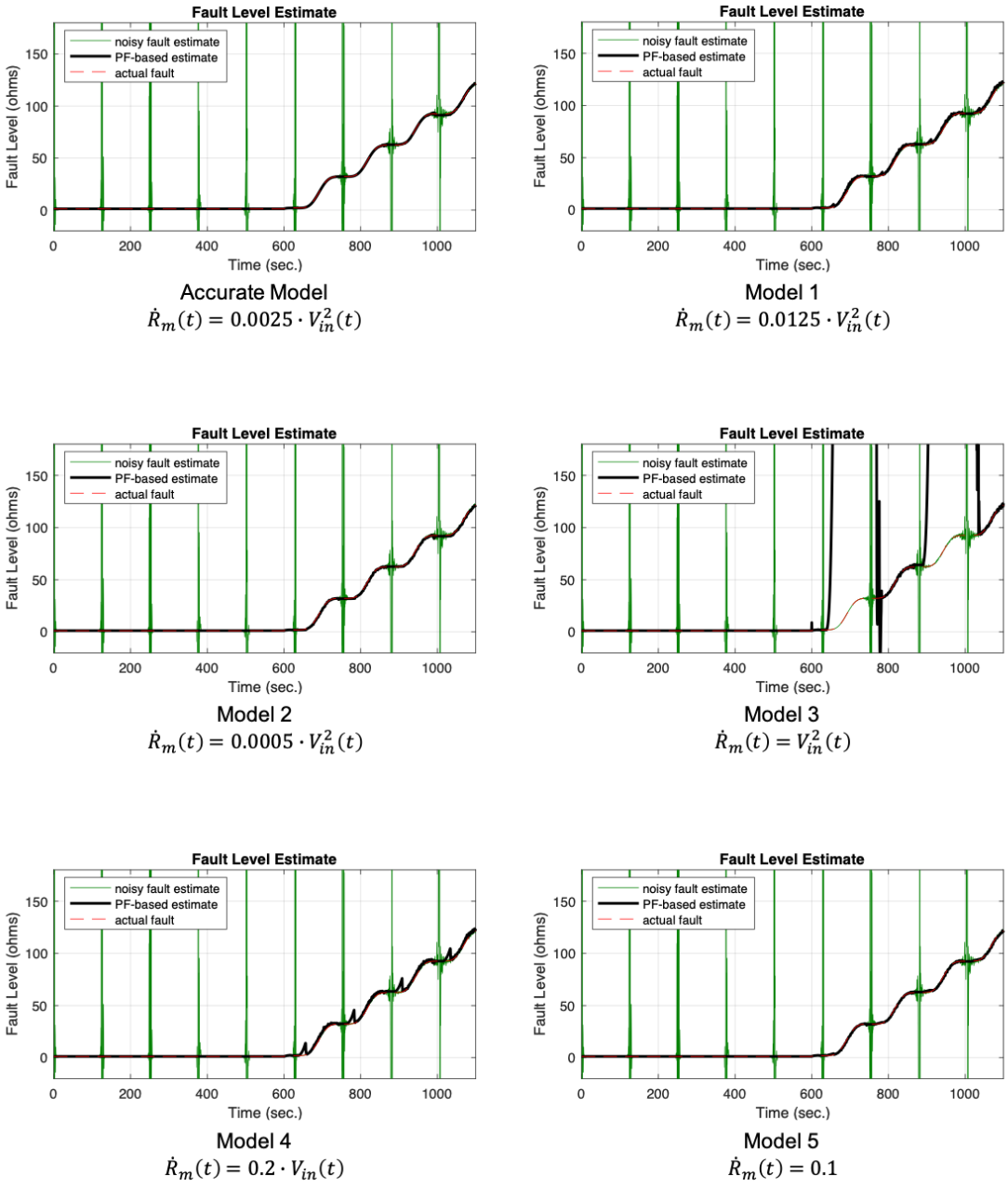

Figure 4.71: Fault state estimation comparisons with respect to fault growth model accuracy

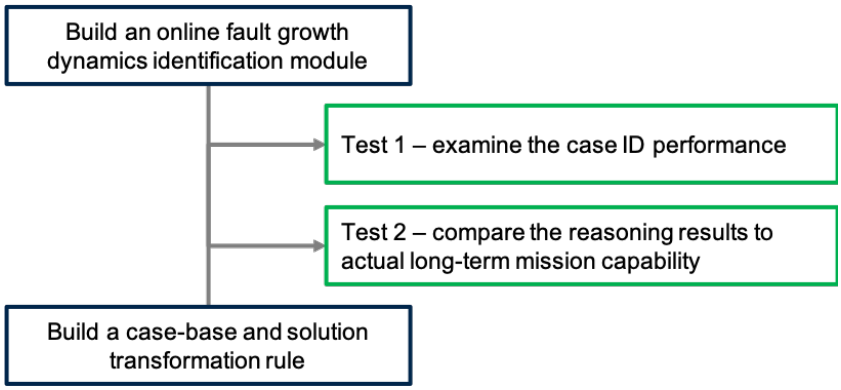

Figure 4.72: Overall procedure and test scenario of Experiment 6

4.8 Experiment 6: Hypothesis test for Module 3 (CBR)

The fault growth model affects not only the fault level estimation but also the long-term mission capability significantly. Therefore, knowing the fault growth model is critical to the adaptation for long-term mission capability in Module 2. If Experiment 6 shows that CBR results in long-term mission capability prediction model, which matches the actual mission capability, then it will support Hypothesis 3-2.

Based on Research Question 3-2 and Hypothesis 3-2, Experiment 6 was designed to test the performance of CBR in Module 3. Three fault growth models were set up for cases in CBR. Then, an online regression module for fault growth model coefficients was developed. Based on the cases and online regression capability, case identification performance was observed. Finally, CBR reasoning results were compared to the simulation data. Figure 4.72 depicts the overall steps of Experiment 6.

- **Research Question 3-2:** How to predict mission capability for Module 2 when a new fault growth model is identified? $\rightarrow$ CBR
- **Hypothesis 3-2:** If a new fault growth model is detected, CBR will be able to produce reasonable mission capability prediction.

Figure 4.73 shows three different simple fault growth models that were included in the case-base. Obviously, with the same control input profile, three different models resulted

in different fault growth patterns. It was assumed that the fault growth model structures are the same for the same fault mode.

Estimation of fault growth model coefficients cannot be deterministic because of measurement and process noises. Considering the stochasticity, cases were formulated based on the statistic model parameters. Gaussian distributions were assumed. Figure 4.74 shows the coefficient distributions for each module.

4.8.1 Test 1: Case identification

The case identification performance was tested in Test 1 based on the cases and online estimation module. Figure 4.75 is the online estimation of the fault growth model coefficient and case detection results. The left and right columns are when $\dot{R}_m = 0.0025 \cdot V_{in}^2$ and $\dot{R}_m = 0.0005 \cdot V_{in}^2$ were simulated, respectively. The first row is the progression of fault states, the second is coefficient estimation history, the third is the coefficient estimation error profile, the fourth, fifth, and sixth are the detection confidence for Case 1, Case 2, and Case 3, respectively. Due to the fault growth model estimation, fault levels were estimated correctly. Also, as shown in the detection confidence, the case identification was correctly made in this example; $\dot{R}_m = 0.0025 \cdot V_{in}^2$ was identified as Case 2, and $\dot{R}_m = 0.0025 \cdot V_{in}^2$ was identified as Case 3.

Note that at the early phase of the coefficient estimation of the fault growth model, the errors appeared larger than later. The fault growth can only be seen after the fault occurred; thus, small sample size at the early phase made significant errors. The errors made the detection delays in the case identification. The impact of this delay needs to be tested in Experiment 7.

4.8.2 Test 2: CBR-based reasoning about long-term mission capability

Test 2 introduced a different coefficient from ones in the case-base. As described in Chapter 3, Shepard's interpolation method was used as a solution transformation rule. Figure 4.76

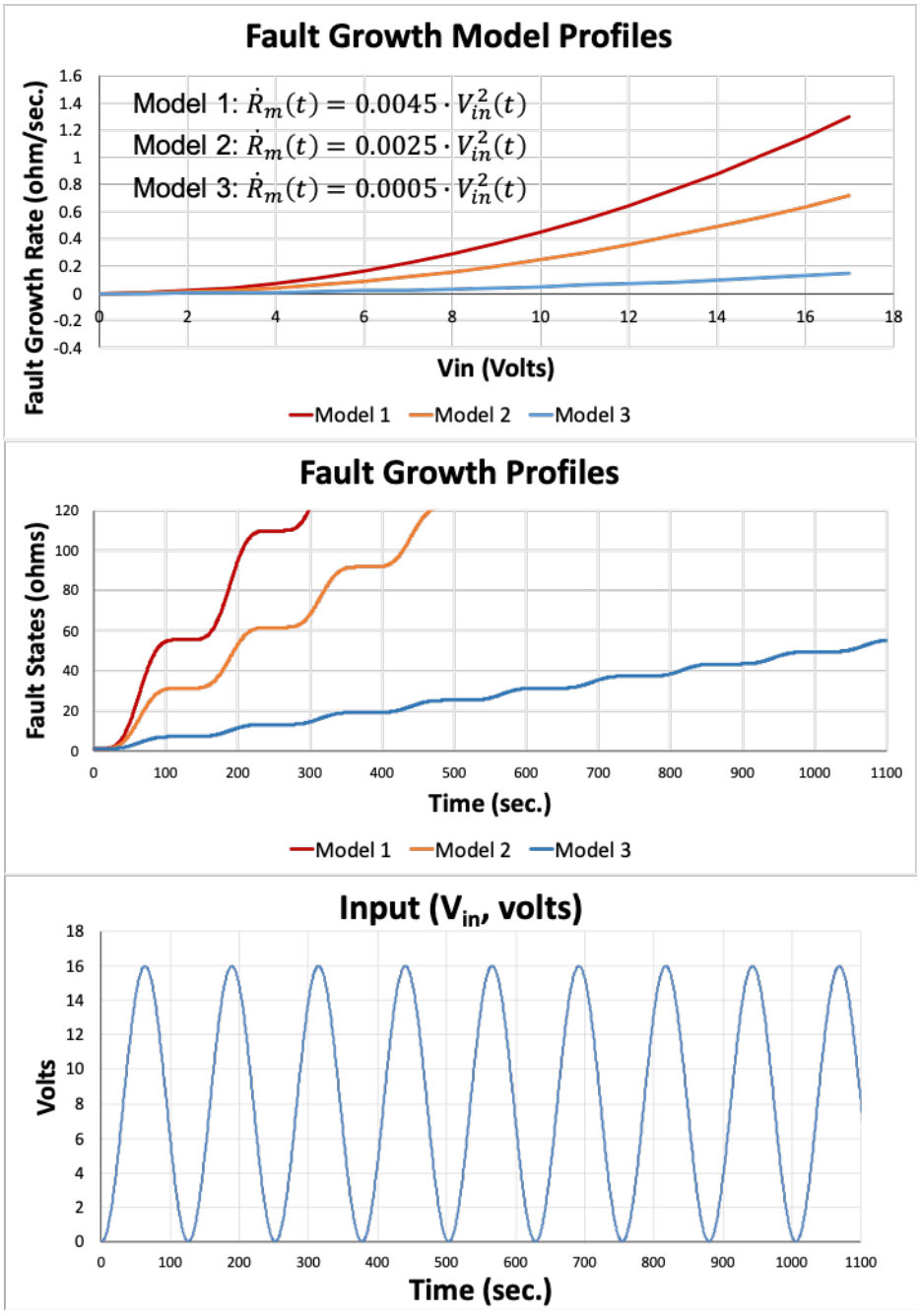

Figure 4.73: Different fault growth models

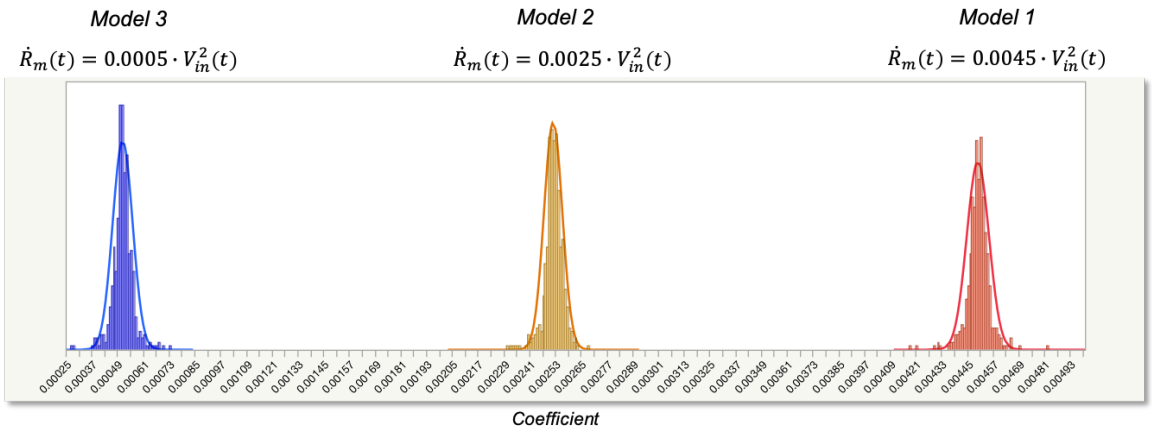

Figure 4.74: Coefficient distributions for each model

shows the comparisons between CBR-based prediction results and simulation data. The induced fault growth model was $\dot{R}_m = 0.0035 \cdot V_{in}^2$. The CBR results indicate the reasonable predictions in terms of the mean and standard deviation of traveling distance available.

4.8.3 Summary of Experiment 6

Results of Experiment 6 supported Hypothesis 3-2. It showed that Case-Based Reasoning could suggest mission capability prediction for a new fault growth model. Three cases were built in the case-base. By the online estimation of the fault growth model coefficient, state levels and case identifications were done successfully. Based on the case identification, if there are no matching cases, CBR reasoned about the long-term mission capability from cases. The observations are summarized below:

- Observations

    1. The online estimation module successfully did the estimation of the fault growth model coefficient, and it helped the case identification.
    2. Large estimation errors of fault growth model coefficient were observed in the early phase of the estimation.

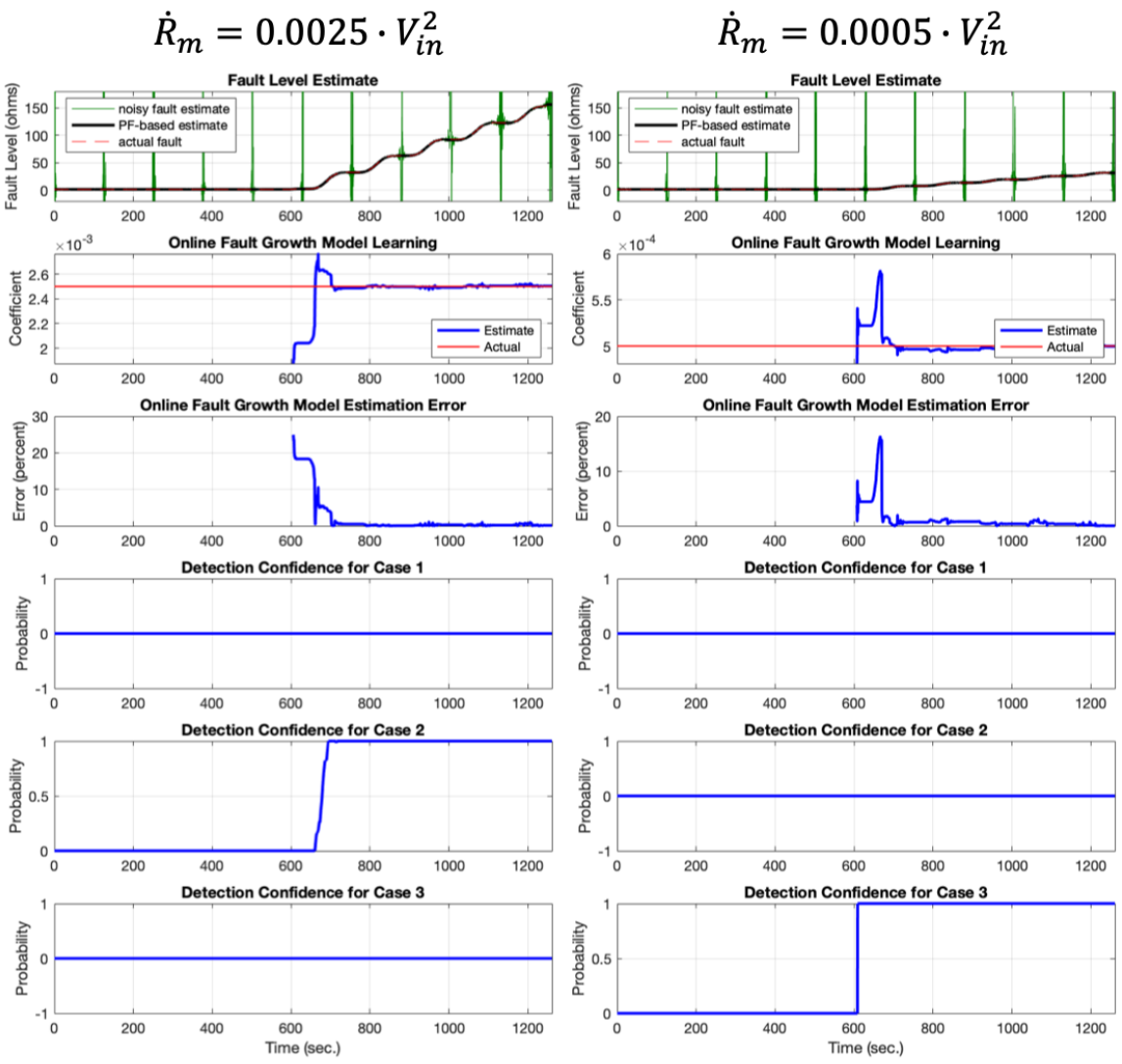

Figure 4.75: Online case identification

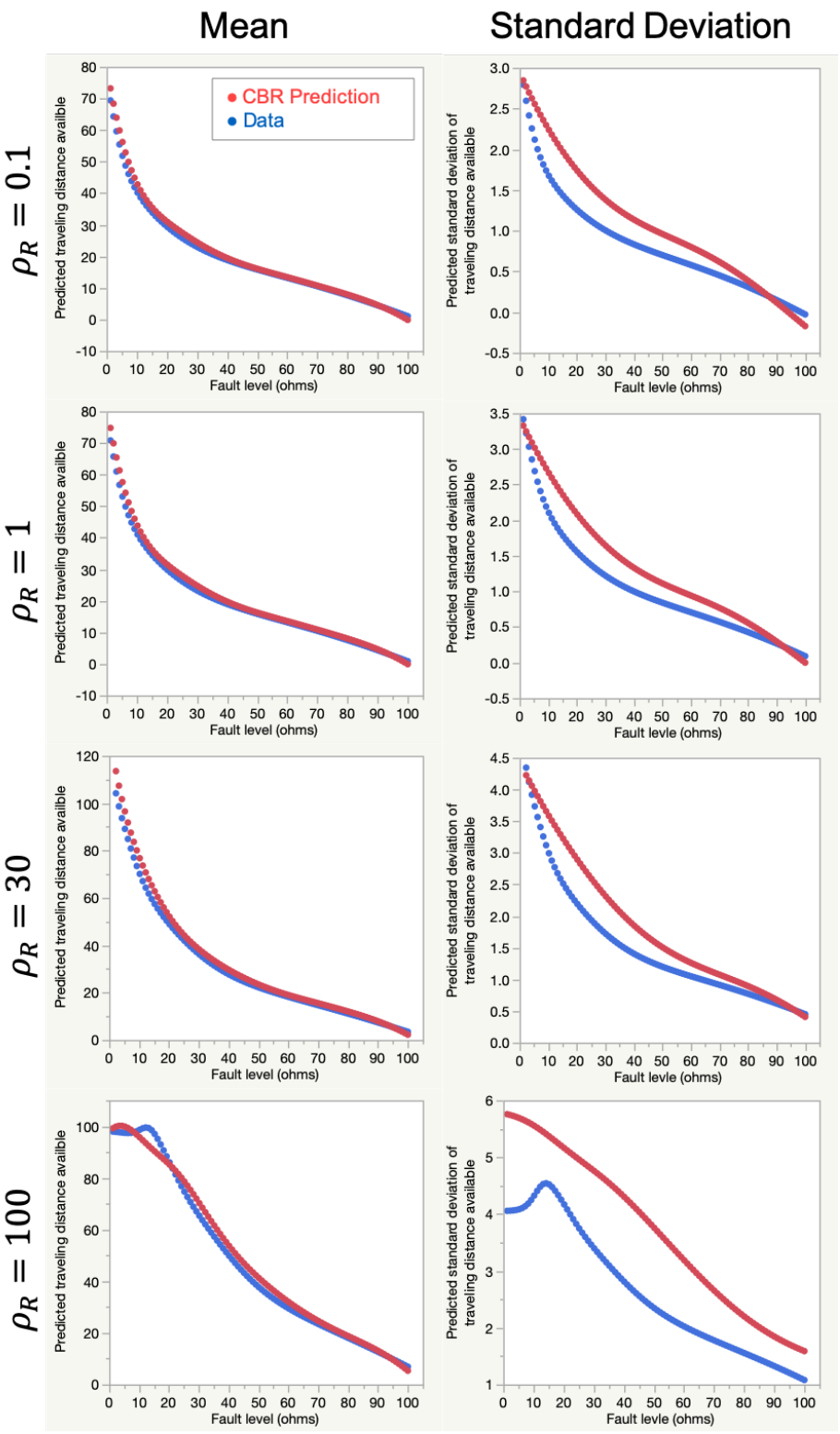

Figure 4.76: CBR Prediction Results by ρ_R: red dots are CBR results and blue dots are simulation data.

3. The estimation errors in the early phase of the estimation delayed the case identification.

4.9 Experiment 7: Hypothesis test for the whole framework

Experiment 7 is the final test set, verifying the efficacy of the integrated framework. If Experiment 7 shows that the proposed resilience-enhanced reconfigurable control framework increases the probability of mission success for severer fault threat levels, then it will support Hypothesis 4 in that the proposed framework improves system resilience in a consequence perspective. According to Research Question 4-1, 4-2, and Hypothesis 4, two test sets were designed: Test 1 was to examine the efficacy of the proposed adaptation by the degradations at the electrical thrust motor insulation, and Test 2 was to observe the computation time as a reference of real-time applicability.

- **Research Question 4-1:** Will the proposed resilience-enhanced reconfigurable control framework improve system resilience?
- **Hypothesis 4:** If the proposed resilience-enhanced reconfigurable control framework is applied, system resilience will be improved in terms of the probability of mission success in the presence of a critical fault.
- **Research Question 4-2:** Will the proposed resilience-enhanced reconfigurable control framework be applied to real-time control application?

4.9.1 Test 1: system resilience improvement

Test 1 induced an artificial fault in the hovercraft example. The mission was a single-waypoint following task. The waypoints were 200 meters. To observe the improvement of the probability of success, the same fault mode (an electrical thrust motor insulation degradation) with the same fault growth model, $\dot{R}_m = 0.0025 \cdot V_{in}^2$, was induced at different locations with or without the proposed adaptation in Monte-Carlos simulations. As

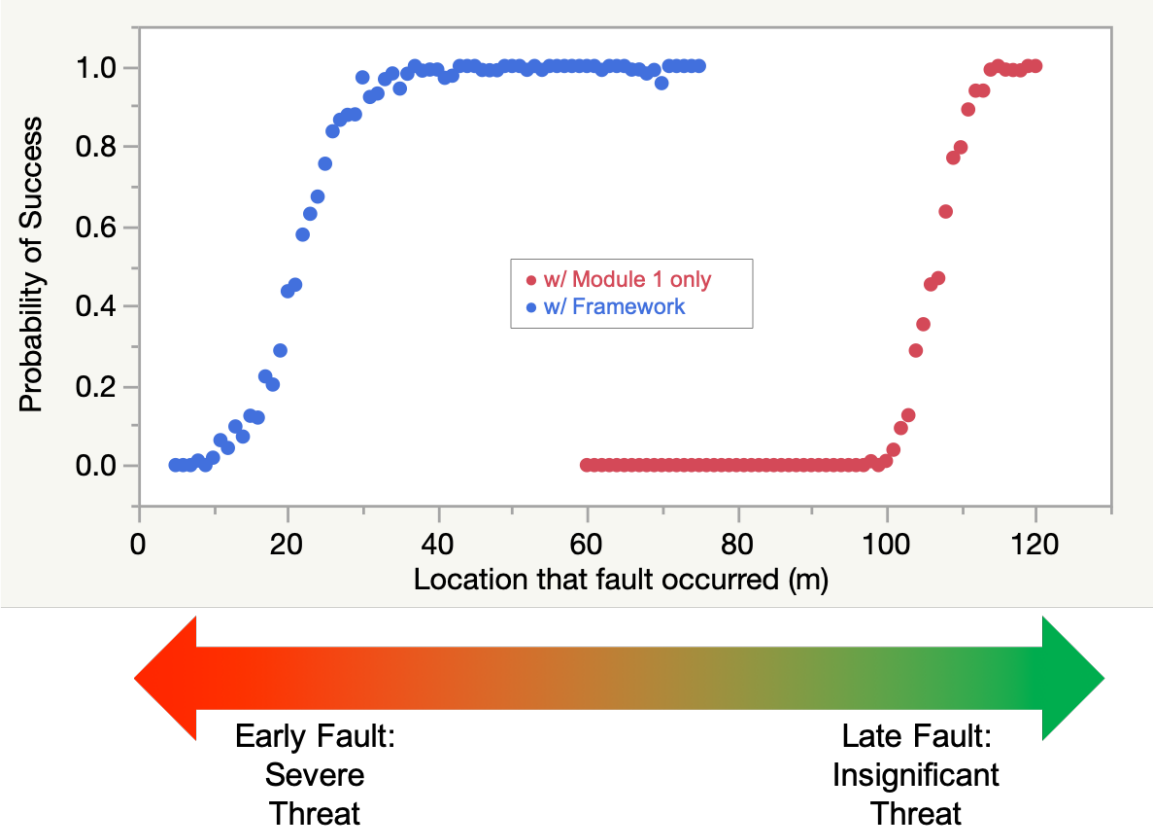

Figure 4.77: Probability of success against the relative position that the thrust fault mode started away from the starting point

mission lengths were the most vulnerable mission capability in this scenario, the initial locations that a fault occurred determined the severity of the fault. Therefore, in this way, the consequences were observed by varying the threat levels in general senses.

Figure 4.77 shows a significant improvement in the probability of success by the proposed reconfigurable control framework. The blue and red dots represent with and without the adaptation. The case without the adaptation maintained the adaptation parameter at 0.1. As a result, the case with the adaptation covered around 80 meters more than the case without the adaptation. It was about 40 percent of the mission length.

Table 4.8: Computation time statistics

Module	Mean (sec.)	Min (sec.)	Max (sec.)
Fault Growth Model Estimation	6.3830e-5	1.0090e-6	0.0034
MPC-DDP	0.0074	0.0062	0.0556
Fault Diagnosis	0.0052	0.0044	0.0116
Adaptation optimization	0.0024	1.0085e-5	0.0382
CBR	0.0021	0.0018	0.0055

4.9.2 Test 2: Computation time

A final piece of the test is to observe the computation time of the entire process. The simulation was performed in Matlab on a regular laptop computer. Computation times of each module were summarized in Table 4.8.

MPC-DDP took the most time on average: 43 percent of the total computation time. The performance of MPC-DDP appeared to drive the total computation time. The maximum calculation time happened at the very first calculations, so it did not affect too much of the entire computation time.

The mean total computational time was 0.0172 sec. Considering the simulation environment, which was a regular laptop and the Matlab, 0.0172 seconds is not a considerable computation time. It would be shortened in a dedicated computation board with an efficient software package.

4.9.3 Summary of Experiment 7

Results of Experiment 7 supported Hypothesis 4-1 by showing that the proposed resilience-enhanced reconfigurable control framework improves the probability of mission success in the presence of a critical fault mode. The improvement of the probability of success showed its effectiveness and potentials of the proposed resilience-enhanced reconfigurable control framework. Observations of Experiment 7 were listed below:

- Observations

Table 4.9: Experiments, hypotheses, and modules

Experiment	Hypothesis	Module
Experiment 1	Hypothesis 1	Module 1
Experiment 2	Hypothesis 2	Module 2
Experiment 3	Test for RL	Module 2
Experiment 4	Test for simulation-based approach	Module 2
Experiment 5	Hypothesis 3-1	Module 3
Experiment 6	Hypothesis 3-2	Module 3
Experiment 7	Hypothesis 4	Framework

1. Probability of success with respect to the severity of a fault mode was improved by the long-term mission capability recovery in Module 2.
2. Computation time shows the potentials of the proposed framework in real-time applications.
3. The longest computation time only happened once at the very first computation for each module. Mainly because initialization took the majority of the time.

4.10 Summary

Chapter 4 described the experimentation plans based on research questions and hypotheses derived in Chapter 3 and tested each module & the integrated framework. The main testbed was under-actuated hovercraft. The introduced critical fault mode was an electrical thrust motor insulation degradation, which causes the loss of thrust effectiveness and control asymmetry eventually. In total, seven experiments were performed. Table 4.9 summarized experiments, hypotheses, and relevant modules of the framework. Test results helped decide the selection between alternative methods and showed the efficacy and applicability of the proposed resilience-enhanced reconfigurable control framework.

CHAPTER 5
DESIGN METHODOLOGY

Chapter 5 describes a design methodology of the proposed resilience-enhanced reconfigurable control framework. Of course, a reconfigurable controller itself is specialized to a given mission and a specific system application, but the design procedure should be generalized so that the proposed framework is designed to improve the resilience of general dynamical system applications.

Figure 5.1 depicts an overview of an offline design procedure by laying out sequential and parallel design steps in the bottom half of the figure. Color shades of the boxes - yellow, blue, and red - represent functional modules, corresponding to the elements in the online operation feedback structure in the top half of the figure. The design procedure comes from an integration of the design procedures of each module.

5.1 Design procedure of each module

The proposed technical approach was built upon a set of assumptions, as described in Chapter 3. Especially, assumptions for knowns define necessary elements that a design procedure has to elaborate for the proposed framework. Table 5.1 revisits the assumptions about *knowns* and categorizes them into which module requires to know them.

5.1.1 Module 1: MPC-DDP

Ultimately, a design of MPC-DDP is to find adequate cost coefficients, K, K_f, and R, which satisfy mission requirements. Let us recall the cost function and constraints of MPC-

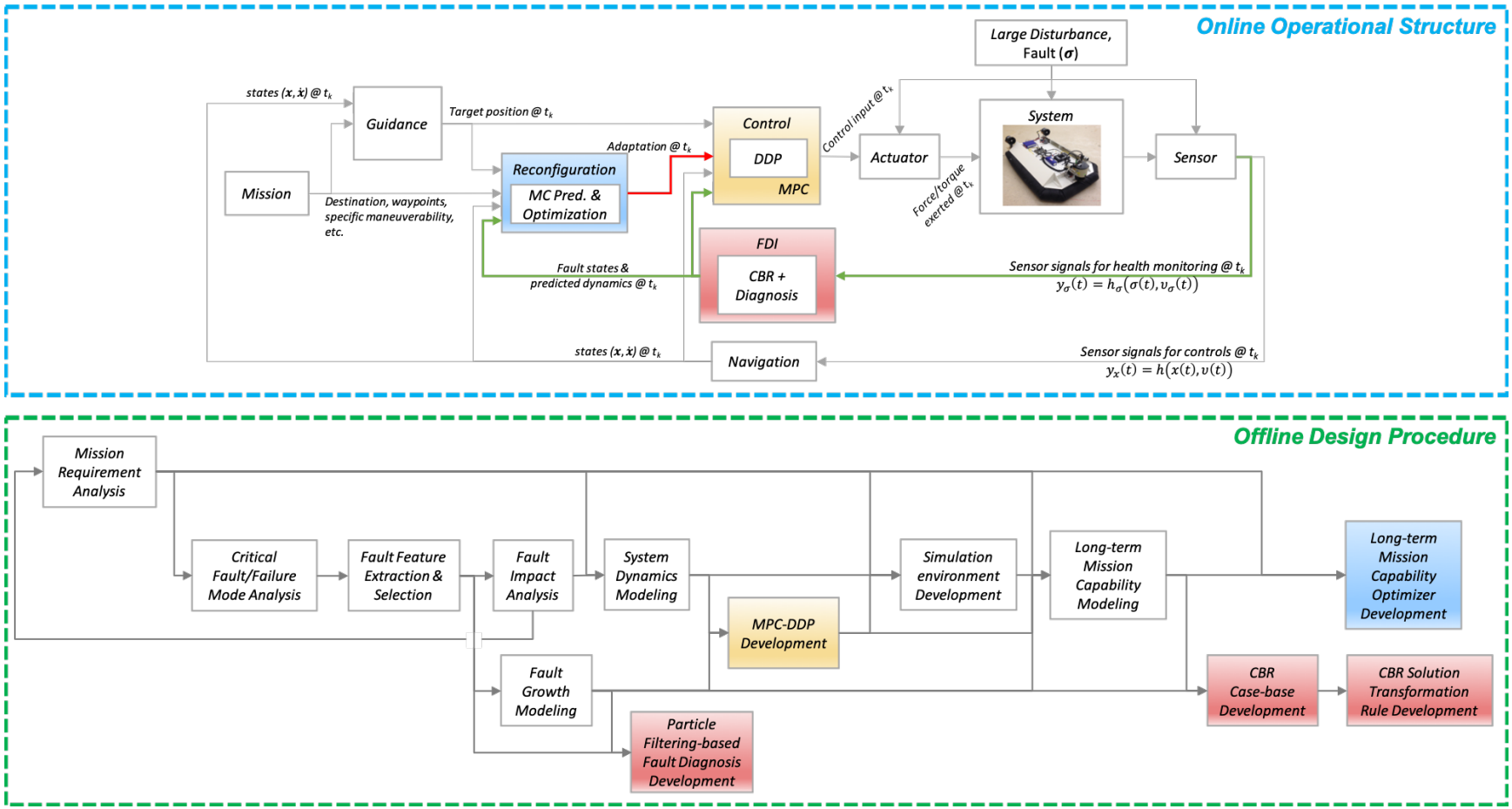

Figure 5.1: Overview of online operational structure and offline design procedure

Table 5.1: Assumptions and relevant modules

Assumptions (Known)	**Categories**
1. Mission and mission requirements are known.	Module 1, 2
2. The most superior mission capabilities are known.	Module 2
3. Critical failure modes and corresponding common fault modes for a given system are identified.	Module 3
4. Vulnerable mission capabilities due to fault modes are known.	Module 2
5. System dynamics models are known and reasonably accurate.	Module 1
6. Fault features are known and selected for each fault mode accordingly. $\rightarrow$ Fault features can isolate a fault mode from others.	Module 3
7. Impact of fault levels on system performances and states are identifiable.	Module 1
8. Typical fault growth models are identifiable.	Module 3

DDP:

$$V\left(x\left(t_0\right),t_0\right)=min_u\left[\int_{t_0}^{t_0+T} l\left(x\left(\tau\right),u\left(\tau\right),\tau\right)d\tau+\Phi\left(x\left(t_0+T\right),t_0+T\right)\right] \tag{3.1 revisited}$$

$$\begin{aligned}\frac{dx}{dt}&=F\left(x\left(t\right),x_f\left(t\right),u\left(t\right)\right)\\ \frac{dx_f}{dt}&=F_f\left(x_f\left(t\right),s_f\left(t\right)\right)\\ s_f\left(t\right)&=F_s\left(u\left(t\right),t\right)\\ g\left(x\left(t\right),u\left(t\right)\right)&\leq 0\end{aligned} \tag{3.2 revisited}$$

$$l\left(x\left(t\right),u\left(t\right),t\right)=\frac{1}{2}\left(x\left(t\right)-r\left(t\right)\right)^T K\left(x\left(t\right)-r\left(t\right)\right)+\rho_R\cdot\frac{1}{2}u\left(t\right)^T Ru\left(t\right) \tag{3.3 revisited}$$

Since MPC-DDP is a model-based optimal control method, system dynamics must be identified before designing a controller. As shown in Experiment 1, fault states and their impact need to be modeled correctly in order to guarantee trajectory recovery in the presence of a critical fault mode. Critical fault modes are assumed to be known, and it will be addressed in a design procedure for Module 3. Then, cost coefficients are tuned based on mission profile, mission desirability, and requirements. This set of design steps for MPC-DDP derives four building blocks:

- Mission requirement analysis
- Fault impact analysis
- System dynamics modeling
- MPC-DDP development (determining K, K_f, and R)

5.1.2 Module 2: Simulation-based long-term mission capability modeling and optimization

From the design procedure for Module 2, an online optimization routine needs to be developed as a function of an adaptation parameter and system/fault states. As defined in Chapter 3, the optimization problem that Module 2 solves is:

$$\max_{\rho_R(t)} J\left(x\left(t\right), x_f\left(t\right), \rho\left(t\right)\right) \tag{3.5 revisited}$$

$$MC\left(x\left(t\right), x_f\left(t\right), \rho_R\left(t\right)\right) \geq MC_{req.} \tag{3.6 revisited}$$

This formulation indicates that the most superior mission capability needs to be determined, and vulnerable mission capabilities due to a critical fault mode should be defined. In order to develop the mission capabilities for the optimization problem, a simulation environment is required. The simulation environment includes system dynamics, fault scenarios, mission profiles, an acceptable adaptation parameter range, and a level of uncertainties/noises. In order to come up with robust predictions, Monte Carlo simulation is performed with discrete levels of an adaptation parameter so that long-term mission capabilities are modeled by statistical parameters. Finally, a long-term mission capability optimizer is developed by an optimization approach. A selection of the optimization approach depends on objective and constraint functions defined. These design steps lead three design steps in addition to the ones for Module 1:

- Simulation environment development
- Long-term mission capability modeling
- Long-term mission capability optimizer development

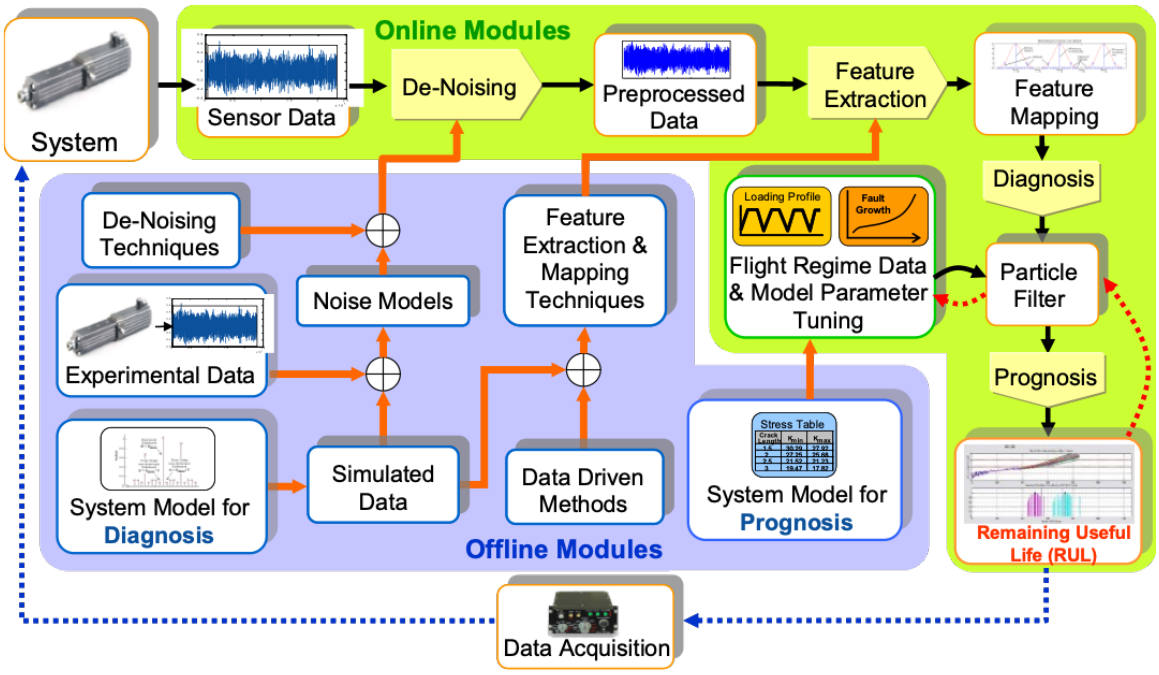

Figure 5.2: Particle filtering-based fault diagnosis and failure prognosis [33]

5.1.3 Module 3: Particle filtering-based fault diagnosis and CBR

Design for Module 3 builds a particle filtering-based fault diagnosis routine and CBR. Figure 5.2 depicts general design steps for a particle filtering-based fault diagnosis. It requires fault modes on interest and their growth models. Based on data or physics-based knowledge, fault features are defined by fault feature extraction and mapping techniques. These knowledge are foundations of the particle filtering-based diagnosis routine.

- Critical fault/failure mode analysis
- Fault feature extraction and selection
- Fault growth dynamics modeling
- Particle filtering-based fault diagnosis routine development

CBR requires to build two elements: case-base and solution transformation rule. Chapter 3 describes how cases are established. By determining knowledge vocabulary, a case

is defined by a problem-solution pair. Assuming that typical fault growth dynamics models are identifiable, model coefficients are signatures of a problem and long-term mission capability is a solution. It derives two building blocks for the design procedure:

- CBR casebase development
- CBR solution transformation rule development

5.2 Design procedure of the resilience-enhanced reconfigurable control framework

The overall design methodology for the proposed resilience-enhanced reconfigurable control framework is a collection of design procedures of each module identified in the previous section.

Step 1: Mission requirements analysis

The mission requirement analysis step helps specify a set of requirements that systems performing the mission must satisfy. It needs to be able to define mission goals, mission constraints, resources, and operating conditions. For complicated systems, Systems Engineering (SE) or Requirement Engineering (RE) disciplines guide systematic approaches and processes as general requirement definition and analysis processes [104].

- Requirement Engineering [104]
    1. Requirements definition and gathering: define top-level requirements and goals
    2. Requirements analysis: refines the top-level requirements such that requirements can be prioritized
    3. Requirements prioritization: handles the conflict of requirements
    4. Requirements flow-down: helps define system and subsystem-level requirements

The proposed resilience-enhanced reconfigurable control framework also requires to define mission requirements when the system is degenerating due to the presence of a critical fault mode during the operation. It may lead to changes in requirements prioritization.

Step 2: Critical fault/failure modes analysis

Critical fault and failure modes need to be identified based on a system specification. Usually, functional and physical decomposition helps reveal each subsystem, component, and their interconnections. Popular tools for health management are Failure Modes and Effects Criticality Analysis (FMECA) [22, 23], Fault Tree Analysis (FTA) [105], Probabilistic Risk Analysis (PRA) [106], or Hazard and Operability Analysis (HAZOP) [107].

As discussed in Chapter 2, the criticality of fault and failure modes can be defined by its inherent severity or impact on system performances and frequency of happening.

Once the critical fault and failure modes are identified and listed, their fault features need to be defined.

Step 3: Fault feature extraction and selection

A fault feature is a processed signal efficiently representing fault characteristics and state of health in a reduced dimension comparing to measurements [25]. Features can be in time, frequency, or time-frequency domains. Fault feature extraction and selection is viewed as a transformation process from data to information and information to knowledge.

Useful fault features have the following attributes [25]:

1. Computationally inexpensive to measure
2. Mathematically definable
3. Explainable in physical terms
4. Characterized by large interclass mean distance and small interclass variance

5. Insensitive to extraneous variables

6. Uncorrelated with other features

There is no generic way of the feature selection method that applies to every fault mode; i.e., the fault selection is application-dependent. However, there are general changes in system properties when a fault occurs in a system [25]:

1. The energy of a system

2. Entropy

3. Power spectrum

4. Signal magnitude

5. Chaotic behavior

6. etc.

There are briefly two ways in fault feature extraction: data-driven and system/parameter identification approaches. Specific methods were described in [25].

Step 4: Fault impact analysis

Since the proposed resilience-enhanced reconfigurable control framework involves a model-based optimal control routine (Module 1), an accurate system dynamics model is required. In this sense, the impact of fault levels on system states and performances need to be identified.

If the fault feature is based on system or parameter identification process, the impact will be easily modeled in system dynamics equations. The hovercraft example in Chapter 4 used such a feature. The actual fault mode was insulation degradation, but the fault feature was represented by armature coil resistance.

If data-driven approaches generate fault features, mapping from the selected features to fault levels and from fault levels to the effects on system dynamics are required. Sometimes, the impact is not much on system dynamics itself, but on the quality of performances such as vibrations or comforts. If the impact is on system dynamics and it is properly modeled, then MPC-DDP module will handle it. If the impact is on the quality of system performances, system performance bounds should regulate the progression of the fault level.

Step 5: Fault growth modeling

Fault growth modeling is one of the critical parts of the resilience-enhanced reconfigurable control design methodology. Fault growth models have significant effects on fault state estimation in a particle filtering-based fault diagnosis routine (Module 3) and long-term mission capability prediction & optimization (Module 2).

Fault growth model can be expressed in a general form as shown in Eq. 3.2:

$$\dot{x}_f = F_f\left(x_f, s_f\right) \qquad \text{(3.2 revisited)}$$

Finding a specific mathematical model of a fault mode can be done by system identification approach. System identification is to find a mathematical model, which fits best to data. System identification uses statistical tools to build input-output relationships from measured data [108]. There are briefly two different approaches:

1. Grey box model: A model structure can be derived by physical relationships between inputs and outputs or expert knowledge. Data determine free parameters.

2. Black box model: Everything is unknown. The black box model approach only relies on data.

As a black box model, symbolic regression can be a promising method for system identification. Symbolic regression finds not only model parameters but also the best model structure [109]. Eureqa is a symbolic regression-based modeling engine [110].

Now, as fault growth models are known, and the fault impact analysis is performed, Step 1 might need to be re-iterated to identify mission requirements and prioritization under the presence of critical fault modes.

Step 6: System dynamics modeling

A system dynamics model is also a cornerstone of the proposed framework. The dynamics model is used in the MPC-DDP optimal control routine online to find optimal control input sequences at every control time. Experiment 1 proved that the accuracy of the system dynamics model has a significant impact on system control performances, and the inaccuracy may even cause catastrophic failure.

System dynamics modeling is often based on physics-based analytical derivations in state space. Model parameters come from system properties and characteristics. Sometimes, however, analytical relationships may not be known. Then, as described in Step 5, the system identification routine should be performed for the system dynamics model again.

Step 7: Particle filtering-based fault diagnosis development

Figure 2.8 illustrates the overview of a particle filtering-based fault diagnosis process. Step 2, 3, and 5 are foundations of Step 7. The theories and analytical derivations were reviewed in Chapter 2 and Chapter 3. A practical design procedure is first to build a baseline for the fault features selected in Step 3. As discussed in Experiment 5 in Chapter 4, the baseline statistics may vary in terms of operating conditions.

Then, a fault level is estimated by the fault growth model, identified in Step 5, and incoming measurements at each estimation time. Statistical thresholds, type-I & type-II errors, determine if the estimate pdf deviates enough from the baseline pdf. The statistical thresholds are design choices.

A basic algorithm of the particle filtering-based fault diagnosis approach is described based on [33]:

Particle filtering-based fault diagnosis algorithm

1. Generate a baseline pdf.
2. Initialize particles, p, weights, w, detection particles, d.
3. Update detection particles based on a random variable
4. Perturb particles based on a pre-defined process noise, fault growth rate, and detection particles.
5. update weights based on measurements.
6. Re-sample weights and particles if particle weights degenerate below a specified threshold
7. Compare the estimate pdf from particles and weights to the baseline pdf, and monitor a statistical measure to declare fault detection.
8. Go to the algorithm number 3, and iterate the process until the operation is finished.

Step 8: MPC-DDP controller development

The theoretical backgrounds were reviewed and described in Chapter 3, Chapter 4, and Appendix C. By the iterative process, control input sequences, u, are obtained for the finite time window based on Eq. 3.1, Eq. 3.2, Eq. 3.3, and Eq. 3.4.

To properly design the MPC-DDP controller, there are design choices for MPC-DDP settings.

1. Size of a finite time window for MPC

2. States and control inputs

3. States references

4. A structure of cost functions

5. Cost coefficient matrices, K, R, and K_f (cost coefficients for terminal states)

6. state constraints

7. control input constraints

8. Intermediate target points for a finite time window (optional)

A finite time window for MPC is the duration of prediction and optimization in MPC. A long time window results in a better optimal control sequence than a short time window but requires more computational time and power. A short time window results in faster solutions for fast applications than a long time window but, the quality of optimality is sacrificed. Therefore, the decision of the time window is system and scenario-dependent. A proper trade-off study would be helpful.

States need to represent the Markov property of the system. The Markov property is a necessary condition for the principle of optimality, which the DDP solution employs. It is said to have the Markov property if a state at a particular time is only dependent on the previous state and action. Almost all systems have the Markov property. However, state settings affect the dynamics model to have or not to have the Markov property. For example, if a helicopter dynamics is modeled by three orthogonal position and velocity vectors, the model has the Markov property. If the dynamics model only has three position vectors, then the next state cannot be determined only be control actions, but all the past position vectors.

The design of K, R, and K_f depends on desirable system performances defined by Step 1: mission requirements analysis. Fault states may be included in the formulation, but

since the long-term mission capability prediction and optimization routine (Module 2) will eventually regulates the fault growth, the inclusion of fault states in MPC-DDP would not be necessary. It is beneficial to have Module 2 because the MPC-DDP formulation will be way simpler than having all possible critical fault states in state and cost function equations.

Hard constraints in states are, unfortunately, difficult to handle in the DDP formulation. Usually, therefore, state constraints are included in the objective function as soft constraints.

Step 9: Simulation environment development

For long-term mission capability prediction modeling, simulation environment needs to be developed. The foundations of the simulation environment are mission (Step 1), critical fault scenarios (Step 2), fault growth models (Step 5), system dynamics models (Step 6), a fault diagnosis routine (Step 7), and an MPC-DDP controller (Step 8.) By putting them all together, system behavior and performances can be tracked and monitored.

Simulation environment, however, can never be the same to the real world because of uncertainty factors. Uncertainty sources and their representations were described in Chapter 4.2. Characterization and quantification of uncertainty for a system and operating conditions are necessary for an accurate simulation environment.

The software simulation environment is also fundamentally different from the real world because the software simulation is discrete. Therefore, a time step size can be another source of harming the accuracy of the simulation. Generally, the small size of the time step results in more accurate results than a large size. However, more computational power is required as the step size becomes smaller.

Another concern is the asynchronous acquisition of sensor data. In other words, obtaining sensor measurements and control time cannot be the same in the real world. Several simulation environments can inherently handle this issue; for example, Robot Operating System (ROS) and Gazebo simulator can simulate virtual sensors and sensor signals just

like hardware sensor measurements. ROS is a middleware handling sensor measurements and software signals [111]. As a middleware, ROS can process both hardware and virtual measurements in the same way. Gazebo is a high fidelity 3D robot simulator [112]. Gazebo uses an open-source physics engine, called Open Dynamics Engine (ODE) [113], which solves 3D rigid-body dynamics numerically. Such simulators can help simulate a system dynamics and behavior more realistically.

Step 10: Long-term mission capability prediction modeling

Step 10 is to identify the maximum capacity of mission capability for the adaptation parameter, ρ_R, by Monte Carlo simulations. For efficient modeling, a proper design of experiments is necessary.

Without loss of generality, long-term mission capability can be affected by four variable sets:

$$MC = F_{MC}\left(x, l_r, x_f, \rho_R\right) \tag{5.1}$$

where x is system states, l_r is the level of future load on a faulty system, a fault state, x_f, and an adaptation parameter, ρ_R. It implies that the simulation experiments need to include:

1. A range of possible states
2. A range of future load on a faulty system
3. A fault state ranging from healthy to failure
4. A range of the adaptation parameter

As discussed in Chapter 3, the adaptation parameter is set constant within a single episode. At each simulation episode, different adaptation parameter values are introduced. For the system and fault states, initial states are controlled at each episode so that the impact of the fault state levels on the long-term mission capability can be easily obtained.

Table 5.2: An example of the table structure for the Monte-Carlos simulation results

x	l_r	x_f	ρ_R	MC_1	MC_2
23	1	1	20	120	89
23	1	1	10	103	78
		$\vdots$			

The future reference trajectory can represent the level of the future load. For instance, a straight reference trajectory will be different from a zig-zag trajectory in difficulty perspectives. Thus, a mapping from the reference trajectory to the load level variable is needed:

$$\text{Future Load Level}\,(t_k) = f\left(x_r\left(t_k, t_{k+1}, \cdots t_{k+N}\right)\right) \tag{5.2}$$

An example result is shown in Table 5.2. In this example, MC_1 and MC_2 are 120 and 89, respectively, if $\rho_R = 20$ at $x = 23$, $l_r = 1$, and $x_f = 1$.

With MC's as dependent variables and others as independent variables, MC's can be modeled by regression. In general form,

$$MC_i = f_{MC_i}\left(x, l_r, x_f\right) \tag{5.3}$$

In nature, introducing higher-order or more regression coefficients will improve the quality of fit. However, the fault growth model should be as simple as possible in this framework. Because, later in Step 12, the regression coefficients will be used as a part of a case in CBR. A large number of coefficients would cause difficulty in case identification in CBR.

Considering uncertainties, MC would be parameterized by a statistical model. For instance, assuming Gaussian distribution assumption,

$$MC_i = f_{MC_i}\left(x, l_r, x_f\right) \sim N\left(\mu_{MC}, \sigma_{MC}\right) \tag{5.4}$$

The statistical model will enable to produce robust solutions in the optimization step.

Step 11: Development of the long-term mission capability optimization routine

The optimization is formulated to maximize the most superior long-term mission capability and to constrain the vulnerable long-term mission capabilities to satisfy the mission requirements. The superior and vulnerable mission capabilities are obtained in Step 1. A selection of the optimization approach depends on objective and constraint functions defined.

$$\max_{\rho_R} J\left(x, l_r, x_f, \rho_R\right) \tag{5.5}$$

subject to

$$MC_i\left(x, l_r, x_f, \rho_R\right) \geq MC_i^{req.} \tag{5.6}$$

where J is the most superior long-term mission capability, and $MC_i^{req.}$ is the i^{th} mission requirements.

In the constraints, statistical models for MC_i can be introduced.

$$MC_i^{\alpha}\left(x, l_r, x_f, \rho_R\right) \geq MC_i^{req.} \tag{5.7}$$

where α is a confidence level and MC_i^{α} is the long-term mission capability limit at α confidence.

Step 12: CBR case-base development

A case is composed of a problem-solution pair. As described in Chapter 3 and Chapter 4, a problem is represented by a vector of regression coefficients of a fault growth model, which was analyzed in Step 5. A solution is a mission capability prediction model identified in Step 10. A case representation was shown in Eq. 3.11 and Figure 5.3.

$$c_i\left(\text{coef.}_i, MC_i\left(x, x_f, s_f\right)\right) \in CB \tag{3.11 revisited}$$

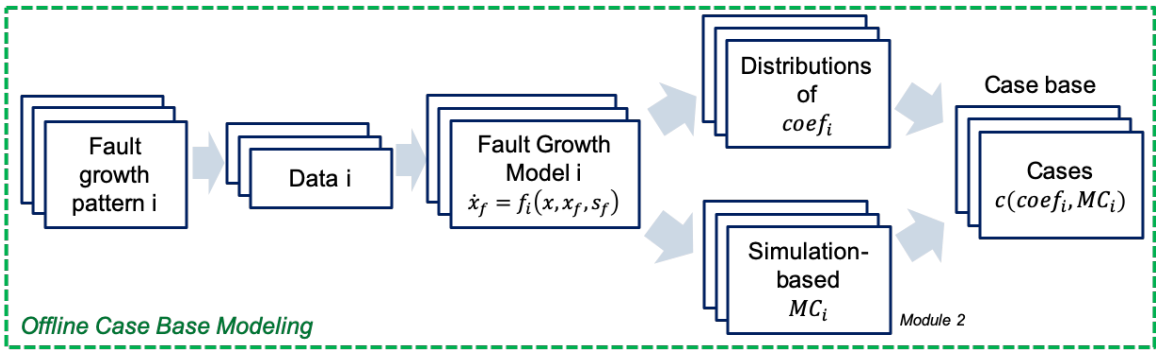

Figure 5.3: Offline case-base modeling procedure

A case-base is a collection of cases. In general, the more cases, the better reasoning results.

Step 13: CBR solution transformation rule development

As described in Chapter 3, a distance-based solution transformation rule can be used by assuming that similar cases result in similar consequences. Since a shorter distance must have more weights, an inverse distance-weighted average method can be applied. Chapter 3 introduced Shepard's interpolation method, but there is no one method to solve every problem. The rule relies on systems, fault modes, and missions.

5.3 Summary

Chapter 5 described a step-by-step design procedure for the proposed resilience-enhanced reconfigurable control framework. A set of assumptions were reviewed as requirements and limitations for the proposed framework, and research problems that the framework addresses. The requirements are included in the design methodology to generalize the framework and the design procedure to work for applicable dynamical systems. Limitations revealed the applicable systems for this reconfigurable control framework.

The design methodology generalized the processes and opened up options. At the same time, the characteristics of each module and their interconnections suggested specific ways to guide for better results out of the framework. The thirteen design steps produce three

necessary elements for the proposed framework. These elements fit into the feedback control structure for the online adaptation and control processes.

CHAPTER 6
CONCLUSION

This chapter concludes the research by reviewing the research objectives, the technical approaches, and test results along with research questions and hypotheses. Research contributions are summarized, and potential future works are described for further advancement based on the fundamental concept of this research.

6.1 Research reviews

This research is motivated by the growing need for safety improvement for unmanned systems in practical applications. Historical incidents have suggested that mechanical failure is one of the key factors threatening the unmanned system safety.

Researchers have been studied for safety improvement in many different aspects. Chapter 2 reviewed the past researches and state-of-the-art techniques. The literature reviews helped identify research gaps that drove this research.

- Research Gap 1 from resilience design approach: A resilience design approach does not explicitly address the practical implementation of adaptation/recovery policies under threats.

- Research Gap 2 from prognosis-enhanced reconfigurable control frameworks: If the mission time required is unknown, current prognostics-enhanced reconfigurable control methods cannot find a control strategy, properly extending RUL over mission time required.

- Research Gap 3 from prognosis in MPC formulation: Performance of MPC-based approach with SOH bounds is very sensitive to the SOH bounds, which are highly dependent on a mission time required and fault growth patterns & rates.

Research Gap 2 and 3 pointed out a lack of a comprehensive long-term mission capability assurance, which is an essential concept of resilience. By hypothesizing that the comprehensive long-term mission capability prediction and optimization will improve the system resilience, the research goal of this research is defined: to develop a design methodology for reconfigurable control framework handling critical fault modes in consideration of system resilience improvement.

By testing and comparing possible alternatives, the proposed resilience-enhanced reconfigurable control framework is composed of three fundamental modules:

1. Reconfigurable controller by MPC-DDP
2. Long-term mission capability optimization by simulation-based mission capability modeling
3. Situational awareness by CBR with repect to different fault growth patterns

Research questions helped shape the technical approaches, and hypotheses derived a set of test cases to prove or disprove the efficacy of the suggested technical methods for each module and the whole integrated framework as well. The main focuses were research questions about the adaptation parameter introduced in the MPC-DDP formulation. Thus, it was hypothesized that the adaptation parameter determines the control strategy influencing long-term mission capability eventually. Another significant problem that this research addressed was the impact of uncertainty in fault growth model on the quality of adaptation. A CBR approach was introduced to properly optimize the adaptation parameter when a new fault growth model was observed.

As a testbed, under-actuated hovercraft with a waypoint-following mission was introduced. An electric thrust motor insulation degradation was artificially induced in simulations as a critical fault mode affecting the overall mission capability. The proposed framework, however, is not limited to the hovercraft application. The assumptions of the

proposed framework described in Chapter 3 and Chapter 5 suggested the necessary properties of applicable systems.

Test results in Chapter 4 showed the resilience improvement of the integrated framework. The introduced adaptation parameter enabled the mission capability trade-offs: mission length vs. mission time in the hovercraft example. The long-term mission capability optimization module (Module 2) was able to find a proper adaptation parameter extending the usable mission length while minimizing the impact of the adaptation to other mission capabilities. The CBR approach for situational awareness improved the system resilience for unknown fault growth patterns.

Finally, the design methodology for the proposed reconfigurable control framework was proposed. For that, a set of assumptions determining required properties for the framework was reviewed. It derived the entire procedure for the proposed resilience-enhanced reconfigurable control framework in general senses.

6.2 Research Contributions

This thesis presents the reconfigurable control framework for system resilience improvement. The contributions of this work are summarized below:

- An introduction to the concept of resilience in AFTCS. This research addressed the necessity of the resilience in AFTCS not only to recover the operating point but also to find a new optimal operating point with the consideration of the long-term mission capability recovery.

- An introduction to the adaptation parameter. This research realized an idea that the *shape* of the objective function changes the optimal operating point by the adaptation parameter. Minimizing the risk of having ill-conditions for DDP, the adaptation parameter was able to adjust the level of regularization of control inputs.

- Consideration of the impact of the adaptation on mission capabilities. An adaptation

not only enables the recovery of vulnerable mission capabilities but also causes sacrifices on other mission capabilities. This research addressed this issue and proposed the mission capability trade-offs in the presence of a critical fault mode.

- Resilience-enhanced reconfigurable control framework. A noble resilience-enhanced reconfigurable control framework was proposed as a feedback controller for dynamical system. Three fundamental components for the framework were established: immediate recovery (Module 1: MPC-DDP), long-term mission capability recovery (Module 2: Long-term mission capability optimization), and situational awareness (Module 3: CBR and particle filtering-based fault diagnosis)

- MPC-DDP as AFTCS. An MPC-DDP optimal controller was introduced and tested as AFTCS. One of the test cases revealed interesting characteristics of DDP, which is the capability of finding optimal controls as long as the system is controllable even though the impact of the fault was not known.

- Simulation-based mission capability modeling. Monte Carlo simulations and regression methods introduced modeling of the long-term mission capability in terms of states and the adaptation parameter.

- CBR for unknown fault growth models. This research addressed significant issues regarding the uncertainty of fault growth patterns that could lead to inaccurate fault level estimation and long-term mission capability prediction. CBR enabled a proper reasoning about the long-term mission capability prediction and, in turn, a better adaptation.

- Under-actuated hovercraft and thrust motor fault example. As a dynamical system, under-actuated hovercraft was introduced as a testbed. An electrical thrust motor fault was artificially induced in the simulation affecting system dynamics and behavior significantly. With the waypoint-following mission, mission length and time were

tracked.

- Demonstration of resilience improvement. The test results showed the improvement of the resilience by the probability of success with the integrated framework. It was compared to the test results without the proper adaptation. The proposed framework also improved the mission time.

- Design methodology for the proposed framework. This research proposed the design methodology of the resilience-enhanced reconfigurable control framework. Assumptions were characterized into requirements and limitations of the framework, and problems that the framework addresses. Requirements were addressed as a part of the design methodology.

- Demonstration of test cases in Matlab.

6.3 Future works

The proposed technical methods and integrated framework showed potentials of the long-term mission capability trade-offs in control reconfiguration. However, there are several conceptual and technical aspects to advance resiliency.

1. Learning capability: Resilience systems learn from experience. CBR forms experiences into cases for future use. However, CBR updates cases manually with human intervention. A Dynamic Case-Based Reasoning (DCBR) approach has been studied for automatic updates on cases, but it requires careful design to apply DCBR in the framework properly. Moreover, when a new experience is observed, making a new case with the mission capability prediction model imposes another difficulty in an accuracy improvement. Regarding learning for the mission capability prediction, an RL approach is still an appealing concept. If general reward settings of RL can be found for the framework and stable learning by value function approximation can be entailed, then RL will be able to enhance the learning capability significantly.

2. Unknown mission profile: A mission profile is often not known. For instance, if a drone is supposed to reach a given destination, but the path is unknown, there will be briefly two challenges to be addressed:

   (a) A considerable amount of uncertainty in mission capability prediction modeling: A large amount of uncertainty may lead to the loss of effectiveness of the long-term mission capability prediction models.

   (b) Pathfinding with obstacle avoidance: MPC-DDP can manage to find an optimal path, but online obstacle avoidance capability would be still required.

3. Generalization of CBR: This research uses CBR to obtain reasonable long-term mission capability prediction when a fault mode is known, and only the fault growth model is unknown. In fact, the beauty of CBR is a reasoning capability about the unknown phenomenon from the experiences, even from different domain knowledge. For instance, if a completely new fault mode occurs and threatens system capability, there should be an adaptation strategy based on the knowledge about different fault modes that have similar effects on long-term mission capabilities. This functionality will improve the system resilience to another level.

4. Link to complex system theory: It will provide a theoretical background for the applicability of the proposed framework to general dynamical systems. It may also be able to define system characteristics that the proposed framework can or cannot apply.

5. Identification of real-time applicability bounds: This research only observed the computation time for the proposed framework. It showed potentials of real-time applicability through relatively fast computation time. However, it is still challenging to answer if the proposed framework would work in real-time application because it all depends on missions, system dynamics, and fault growth rates. If particular

conditions or bounds can be defined for those scenarios that the proposed reconfigurable control framework can apply, then it will add great practicality to the proposed framework.

Appendices

APPENDIX A

LITERATURE REVIEWS ON DESIGN APPROACHES FOR SYSTEM SAFETY IMPROVEMENT

The increasing trend of system complexity involves risks in terms of system safety. In order to reduce risks by improving system safety, governmental agencies have recognized the importance of system safety requirements. Department of Defense (DOD) has defined and keeps revising system safety requirements while identifying hazards and risks [114]. The demanding system safety requirements have been pursued by introducing advanced design methodologies, which aim at improving system survivability, system reliability, and risk mitigation [115]. Naval engineering community also highlighted the significance of system effectiveness and survivability as design attributes, and pointed out that those design attributes should be considered in the early design phase [116].

As a matter of fact, traditional conceptual/preliminary design approaches more focused on mission capability, not system safety [117]. The way they improved system safety was to address system certification and safety-related issues after conceptual/preliminary designs. This approach naturally caused more weights and costs because of additional hardware redundancies, design changes, or design iterations, implying less affordability of a system. Specifically, for complex systems, design changes or retrofits compensating for safety issues is not plausible [118]. Figure A.1 notionally describes how severe the late safety implementation would cost in the long-term system utilization perspectives as well as how difficult system safety consideration could be in the early design phases. This indicates that there is research opportunities so as to move safety from retrofit safety enhancement to safety through design approaches. Aerospace and naval authorities and experts also highlighted the importance of early consideration of system safety for military system acquisition [119].

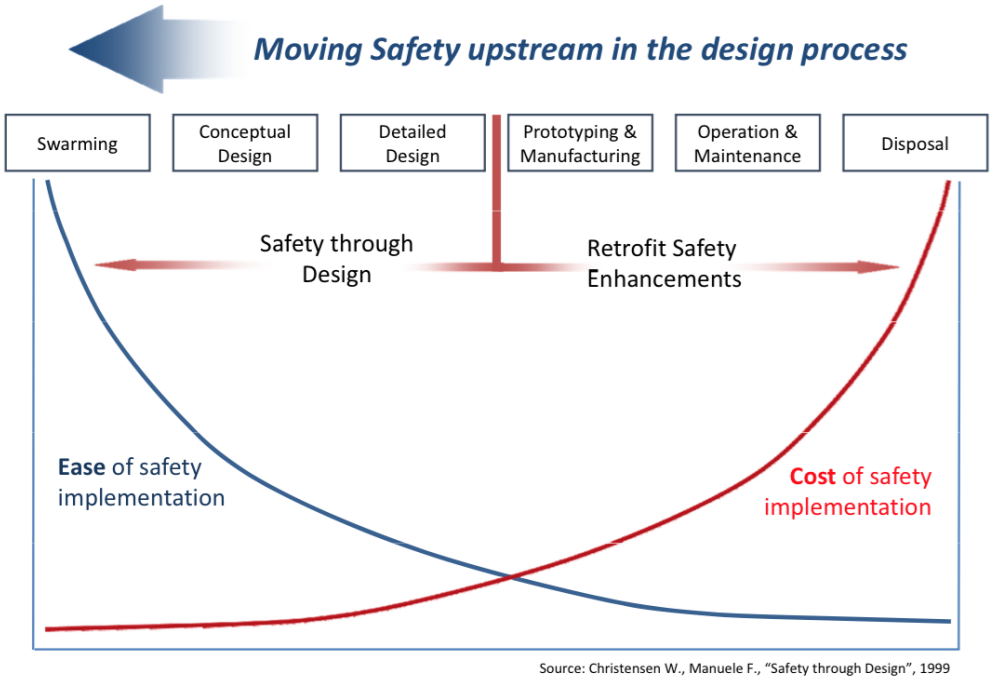

Figure A.1: Safety early in the design process [117].

Major elements of safety management concepts associated with fault events are system reliability and survivability. Reliability is the probability that a system component performs its functionality for a predefined operational time under a specified operating conditions [120]. It is assumed that the effects of maintainability and a rate of deterioration are also factors that assess system reliability. Reliability analysis is a bottom-up approach, evaluating component failures and their effects on system functions. Safety analysis, on the contrary, is a top-down approach. It assesses how dangerous conditions can happen during the operations. It implies that system safety is a super set of system reliability [46]. High reliability, therefore, does not necessarily guarantee a safe system. System safety should be evaluated in a large picture as components operate together in a system [121].

Survivability, on the other hand, is a concept, which was originated from military aerospace applications [122] and populated in other engineering disciplines [123, 124]. As depicted in Figure A.2, survivability consists of susceptibility, vulnerability, and recoverability [46]. Susceptibility is the probability that the system experiences a direct hit

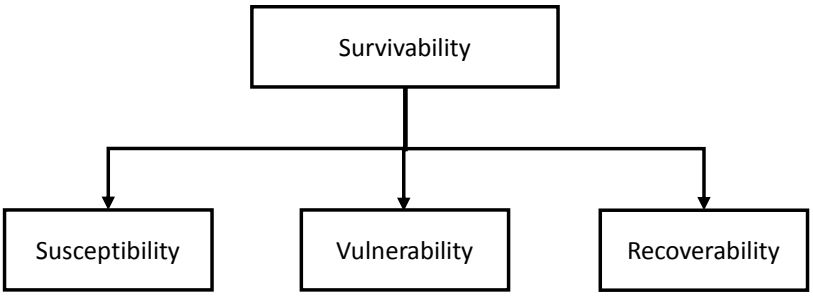

Figure A.2: Survivability breakdown [46].

or secondary hit effects through an attack by its environment [125]. This property differentiates survivability from reliability in that survivability encapsulates system resistance characteristics against externally perturbed system conditions. Vulnerability is defined as the conditional probability that the system is terminated *after* an attack by its environment [125]. Likewise, recoverability is the conditional probability that the system is recovered after an attack by its environment *if* the system is not terminated [125].

Based on the concepts of system survivability and reliability, safety engineers target to confirm and verify whether a system design is safe as a part of system characteristics. Usually, they conduct fault and failure analysis in order to propose safety requirements in the design specifications [46]. The safety management methods have been addressed under risk identification [126], assessment and estimation [127], reliability engineering [128], and survivability engineering [125]. Recently, resilience engineering has been accentuated for safety management [38].

A.1 Risk management

For safety management, risk is an important metric [129]. Risk has been used as many different kinds in many different disciplines [130]: business risk, social risk, economic risk, safety risk, investment risk, political risk, etc. As a basic notion of risk, *Kaplan* defined risk based on the distinction between risk, uncertainty, damage, and hazard [130] as shown in the symbolic equations below:

$$risk = uncertainty + damage$$

$$risk = \frac{hazard}{safeguards}$$

The first equation above pointed out that the notion of risk is a combination of uncertainties and losses as its consequences. The second equation implies that hazard is *a source of danger* and the notion of safeguards is *awareness of risk* [130]. With this basic concepts and distinctions in mind, risk is measured probabilistically based on a list of undesirable scenarios, associated with their consequences [130].

Based on the definition and measure of risk, complex systems inevitably involve more risk. Complexity itself is a critical feature of uncertainties in design and operations [131]. In most cases, complex systems behave nonlinearly and non-deterministically [132]. Challenging mission expectations or changes in operational environment also cause more risks [40].

Risk identification is a popular technique for safety management [46] along with risk mitigation. Risk identification aims at finding the root cause of system faults and failures, which potentially trigger dangerous situations as well as reducing the number of root causes of severest risk factors. With this context, common risk identification techniques are listed below [46]:

- Common cause failure (CCF) analysis [133]
- Fault tree analysis (FTA) [134, 135]
- Failure mode and effect analysis (FMEA) [22, 23]

These techniques concentrate on accident mechanism investigation since most accidents and malfunctions have been caused by performance losses and security & system integrity under the presence of threats and operational hazards [46]. Thus, they develop

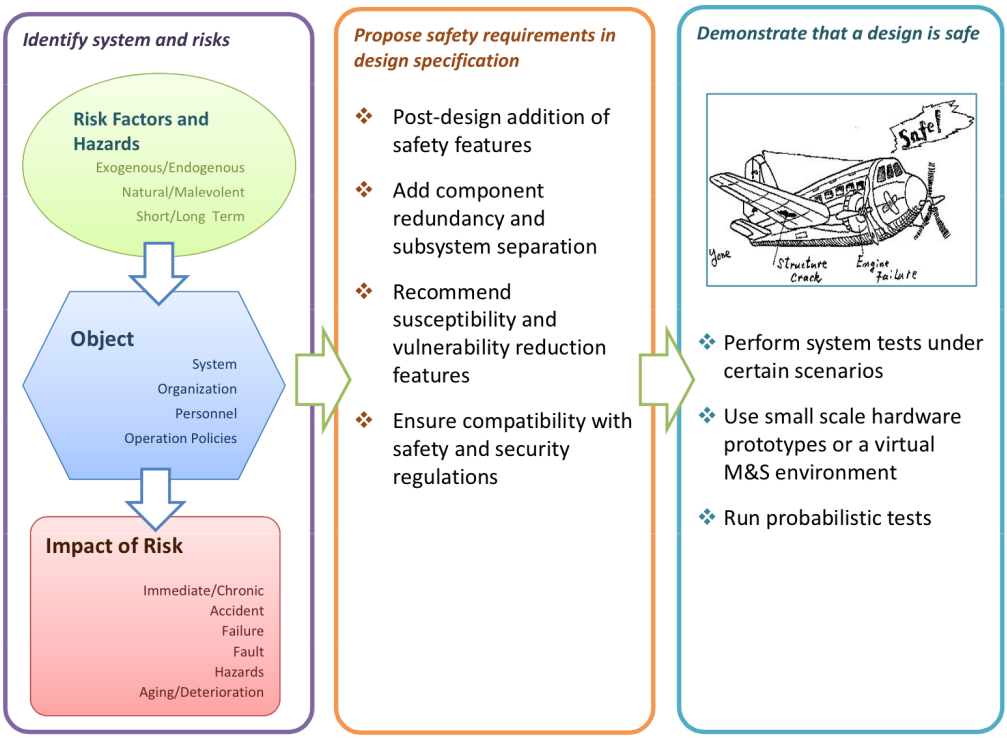

Figure A.3: General safety by-design practical procedure [46].

fault generation and propagation models. Advanced methods have endeavored to develop physics-based modeling and simulation for more accurate accident and damage prediction [136].

In order to make the risk identification techniques effective, critical scenarios should be carefully investigated in a finite set of undesirable scenario list, and policy-making decisions should be carried out for appropriate technology infusion as a means of worst case prevention [137]. As a part of design problem, maximum threshold for risk is identified. Risk assessment is critical for strategic decision support; e.g., hardware redundancy, separation, safety-related technology infusion, etc. Then, risk-benefit studies and prototype testing verify how safe a system is and how further the system safety should be improved. Figure A.3 depicts an overview of current practical procedure in safety by-design approach [46].

A.2 Reliability engineering

Reliability engineering mainly focuses on failures and failure rates reduction, which are root causes of safety-threatening accidents [46]. Failure scenarios can be component failure, incorrect maintenance, control problems, human mistakes, design errors, and their combinations [138]. Thus, reliability engineering aims at minimizing the rate of component failures by functional redundancy, standby sparing, safety margins, and maintenances [139]. One of the famous design techniques is *Reliability-based Design Optimization* (RBDO). It focuses on probabilistic evaluations in system reliability in consideration of known operating environment and uncertainties [128]. Another popular technique is *Robust Design*, which is aiming at improving system quality and performances with various external factor [140].

A.3 Survivability engineering

As discussed earlier, survivability is a key feature of system dependability, especially when other "-ilites" such as reliability, availability, maintainability, etc. - which are also elements of system dependability - do not dynamically change along time. Due to this reason, system survivability has been primarily highlighted as a means of concurrently maintaining availability and capability for more effective systems. Originally, it was a popular concept in the military combat systems. *Ball* summarized combat losses and loss rates for various aircraft at a variety of conflicts from the historical data [125]. This brought survivability-based design philosophy for military systems.

The concept of survivability is also applied to civilian systems because civilian systems also sometimes undergo dangerous conditions, potentially causing accidents; thus, civilian systems should be able to endure the effects of dangerous environment and continue its mission with safe system conditions including crew and passengers. For this, system survivability is a crucial priority [46].

The traditional survivability was formulated as below [125]:

$$P_S = 1 - P_K \tag{A.1}$$

where, P_S is the probability of survival and P_K is the killability or probability of not surviving the disturbance.

Survivability often refers to the resistance against external attacks, Equation A.1 can be expressed as [125]:

$$P_S = 1 - (P_H \cdot P_{K/H}) \tag{A.2}$$

where, P_H is the probability of being detected (susceptibility) and $P_{K/H}$ is the conditional probability of not surviving the attack and getting killed after detection and attack.

Sometimes the formulation of survivability depends on the sequence of events, which are about threats and consequences. In military systems, a *kill chain* was introduced as system survivability assessment support for a system-threat encounter scenarios [125]. According to this kill chain, Equation A.1 can be reformulated with a multiplication of conditional probabilities which represent each chain element in the kill chain model [46]. An important thing to note is that based on the described concept and probabilistic formulation, survivability is scenario and system dependent; thus, applications of survivability are varying in many scientific and engineering disciplines within their own context [46]. Development of general characteristics of survivable systems, though, attempted by *Ellison* in three key notional terms: *essential functions*, *attacks & failures*, and *timely manner* [141]. He defined survivability as "the ability of a network computing system to provide *essential services* in the presence of *attacks and failures*, and recover full services in a *timely manner*." Essential functions are the requirements that a system must deliver. These functions are not only related to mission goals, but also safety-mode operations. Attacks & Failures are undesirable conditions that a system must prevent in order to protect its essential func-

tion. Disturbances are a form of any undesirable interruptions on essential functions such as attacks, faults, or failures. All systems have, on some levels, a natural absorbability with respect to disturbances, but the question will be how absorbable a system is. Lastly, timely manner reflects how responsible a system is against disturbances [46].

Richard extended the definition of system survivability based on the three key characteristics as [42]:

"the ability of a system to minimize the impact of a finite-duration disturbance on value delivery, achieved through (I) the reduction of the likelihood or magnitude of a disturbance, (II) the satisfaction of a minimally acceptable level of value delivery during and after a disturbance, and/or (III) a timely recovery."

He also illustrated a notional behavior of dynamic systems that encounter and withstand disturbances with the introduction to time periods for behavior categorization as shown in Figure A.4 [42].

Epochs are time periods which characterize dynamic system behavior from original states to recovered states. They are fixed properties: i.e., static constraints, design philosophy, technology selection and properties [142]. Epoch 1 is a time period of normal operating conditions at which a system delivers a notional performance value, V_o, as its original states. At epoch 2, disturbances occurs; it causes system performance degradation by a certain level in the end. After epoch 2, a system endeavors to recover so as to make the performance value to be the original state value, V_o within epoch 3. However, a system may not be able to fully recover, or it may not be necessary to make the performance value to be V_o just for survival. Then, a system restores the performance value just to be higher than the minimum value, V_x, by the recovery time, T_r [142].

As for survivability-based design philosophy, many different approaches have been developed for different applications [46]. One of inclusive methods is *Ball*'s method which

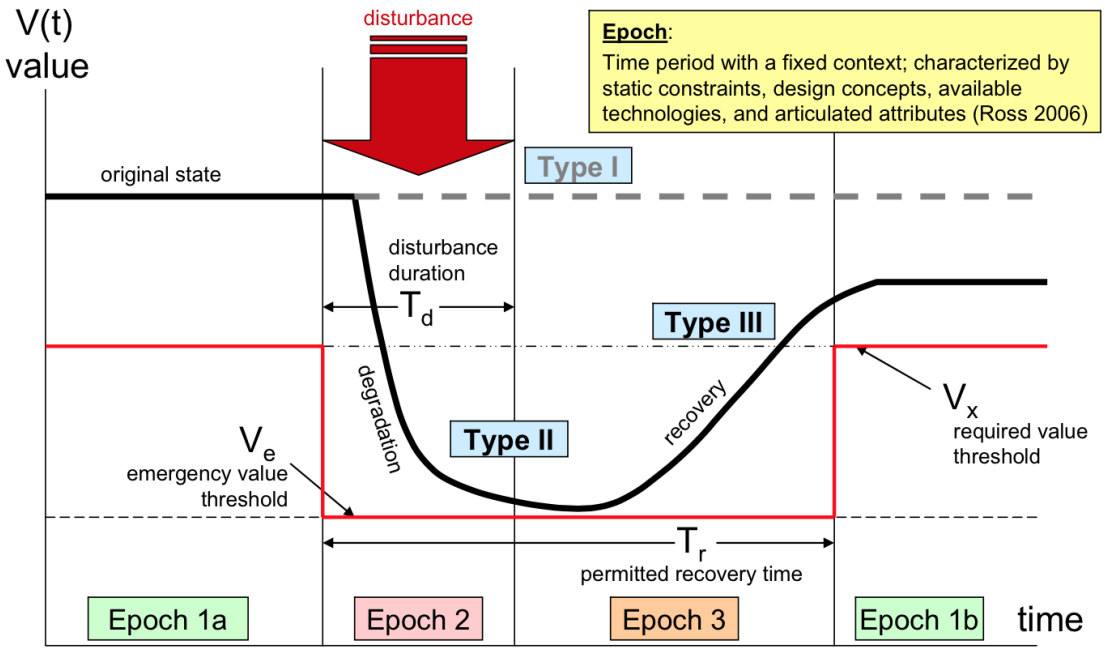

Figure A.4: Conceptualization of survivability [42].

assesses survivability with susceptibility and vulnesrabitliy reduction concepts [125]. He targeted to maximize *Return On Investment* (ROI) by maximizing benefits of survivability while minimizing acquisition and operating costs associated with costs for upgrades [125]. Since it is a design process, design revision-evaluation iterations are necessary. Figure A.5 depicts an overview of *Ball*'s survivability-based design procedure [125]. An example of susceptibility and vulnerability reduction strategies for F/A-18 is shown in Table A.1 [125].

Other entities also developed and suggested survivability-based design methods for their own applications. *Joint Technical Coordinating Committee for Aircraft Survivability* (JTCG/AS) published the Aerospace Systems Survivability Handbook Series in order to provide survivability requirements and suggest survivability considerations in Integrated Product and Process Development / Integrated Product Teams (IPPD/IPT) [143]. The U.S. Navy also promoted survivability as a part of design discipline by offering a standard procedure for survivable ship design and assessment [144].

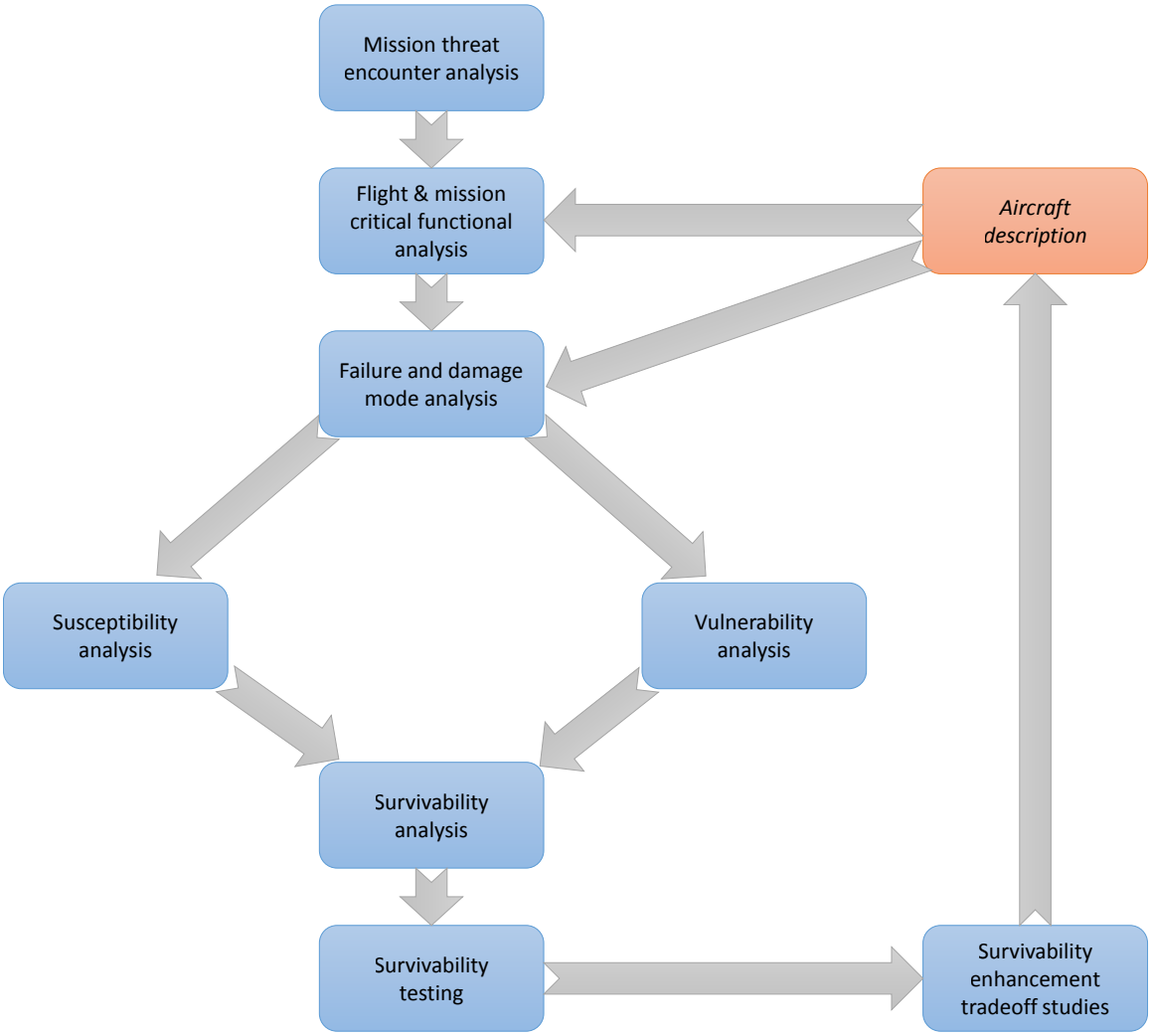

Figure A.5: Overview of *Ball*'s survivability-based design procedure [125].

Table A.1: Susceptibility and vulnerability reduction strategies for F/A-18 [125]

Susceptibility reduction	**Vulnerability reduction**
Threat warning	Component redundancy with separation
Noise jamming and deceiving	Component location
Signature reduction	Passive damage suppression
Expendable	Active damage suppression
Threat suppression	Component shielding
Weapons and tactics, flight performance	Component elimination
Crew training and proficiency	Component replacement

APPENDIX B

STABILITY ANALYSIS FOR MPC

A value function of an optimal control problem in discrete time is:

$$V_N\left(x\left(t_0\right), u\right) = \sum_{j=0}^{N-1} l\left(x\left(t_j\right), u\left(t_j\right)\right) \Delta t + V_f\left(x\left(t_N\right)\right) \tag{B.1}$$

where $x\left(t_0\right)$ is the current state, u is the control input sequence, and $V_f\left(x\left(t_N\right)\right)$ is the terminal cost. The cost function $V_N\left(x\left(t_0\right), u\right)$ evaluates costs for N discrete time segments at $x\left(t_0\right)$. Define $u^*\left(x\left(t_0\right)\right)$ the optimal control input sequence given the initial condition, $x\left(t_0\right)$, and $V_N^*\left(\cdot\right)$ is the costs along the optimal control input, $u^*\left(\cdot\right)$ at any given current state. Considering discrete system dynamics constraints,

$$x\left(t_{k+1}\right) = f\left(x\left(t_k\right), u\left(t_k\right)\right) \tag{B.2}$$

where $u\left(t_k\right) \in U$, U is a feasible input set, and $V_N^*\left(\cdot\right)$ can be said as a Lyapunov function if,

$$V_N^*\left(x\left(t_{k+1}\right)\right) - V_N^*\left(x\left(t_k\right)\right) \leq 0 \tag{B.3}$$

for all t_k. Then, MPC with the cost function, Eq. B.1, is stable. The proof is following:

Suppose that there exist the optimal control input at time, t_k:

$$u^*\left(x\left(t_k\right)\right) = \left[u^*\left(t_k\right), u^*\left(t_{k+1}\right), \cdots, u^*\left(t_{k+N}\right)\right] \tag{B.4}$$

Now, consider the control input sequence at time, t_{k+1} as:

$$u\left(x\left(t_{k+1}\right)\right) = \left[u^*\left(t_{k+1}\right), u^*\left(t_{k+2}\right), \cdots, u^*\left(t_{k+N}\right), u\left(t_{k+N+1}\right)\right] \tag{B.5}$$

Equation B.5 is not an optimal control sequence because $u\left(t_{k+N+1}\right) \in U$ is not optimal. By the definition of the cost function,

$$V_f\left(x\left(t_{k+N+1}\right)\right) + l\left(x\left(t_{k+N}\right), u\left(t_{k+N}\right)\right)\Delta t \leq V_f^*\left(x^*\left(t_{k+N}\right)\right) \tag{B.6}$$

Therefore,

$$V_N\left(x\left(t_{k+1}\right), u\left(x\left(t_{k+1}\right)\right)\right) \leq V_N^*\left(x^*\left(t_k\right)\right) - l\left(x\left(t_k\right), u^*\left(t_k\right)\right)\Delta t \tag{B.7}$$

Finally,

$$\begin{aligned} V_N^*\left(x^*\left(t_{k+1}\right)\right) &\leq V_N\left(x\left(t_{k+1}\right), u\left(x\left(t_{k+1}\right)\right)\right) \\ &\leq V_N^*\left(x^*\left(t_k\right)\right) - l\left(x\left(t_k\right), u^*\left(t_k\right)\right)\Delta t \end{aligned} \tag{B.8}$$

Equation B.3 is proved. (The proof was from [78].)

APPENDIX C

DIFFERENTIAL DYNAMIC PROGRAMMING

C.1 Part I: DDP derivation

Consider optimal control problem with the cost function

$$V(\mathbf{x}(t_0), t_0) = \min_{\mathbf{u}} \left[\phi(\mathbf{x}(t_f), t_f) + \int_{t_f}^{t_0} l(\mathbf{x}, \mathbf{u}, t)\, dt \right] \tag{C.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ the state and $\mathbf{u} \in \mathbb{R}^m$ the control.

Bellman's principle is shown in Equation C.2

$$V(\mathbf{x}(t_k), t_k) = \min_{\mathbf{u}(t_k)} [L(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) + V(\mathbf{x}(t_{k+1}), t_{k+1})] \tag{C.2}$$

where $dt = t_{k+1} - t_k$, $L(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) = l(\mathbf{x}, \mathbf{u}, t)\, dt$ the running cost, and $V(\mathbf{x}(t_k), t_k)$ the value function at state $\mathbf{x}(t_k)$ & time t_k.

C.1.1 Linearize the dynamics in discrete time

A general dynamics is shown as:

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}, \mathbf{u}, t) \tag{C.3}$$

Then, following derivation holds:

$$\begin{aligned} \frac{d\mathbf{x}}{dt} &= f(\mathbf{x}, \mathbf{u}, t) \\ &= f(\mathbf{x} - \bar{\mathbf{x}} + \bar{\mathbf{x}}, \mathbf{u} - \bar{\mathbf{u}} + \bar{\mathbf{u}}, t) \\ &= f(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}, t) \\ &= f(\bar{\mathbf{x}}, \bar{\mathbf{u}}, t) + f_x\delta\mathbf{x} + f_u\delta\mathbf{u} \,(\text{Tayler series expansion}) \end{aligned} \tag{C.4}$$

where f_x and f_u are partial derivatives of $f(\mathbf{x}, \mathbf{u}, t)$ with respect to $\mathbf{x}$ and $\mathbf{u}$, respectively.

Since $f(\mathbf{x}, \mathbf{u}, t) = \frac{d\bar{\mathbf{x}}}{dt}$,

$$\begin{aligned} \frac{d\mathbf{x}}{dt} - \frac{d\bar{\mathbf{x}}}{dt} &= f_x\delta\mathbf{x} + f_u\delta\mathbf{u} \\ \frac{d\delta\mathbf{x}}{dt} &= f_x\delta\mathbf{x} + f_u\delta\mathbf{u} \end{aligned} \tag{C.5}$$

By Euler method, Equation C.5 can be discretized as below:

$$\begin{aligned} \delta\mathbf{x}(t_{k+1}) &= \delta\mathbf{x}(t_k) + \frac{d\delta\mathbf{x}}{dt}|_{t_k} dt \\ &= \delta\mathbf{x}(t_k) + (f_x\delta\mathbf{x}(t_k) + f_u\delta\mathbf{u}(t_k))\, dt \\ &= (I_{n\times n} + f_x dt)\,\delta\mathbf{x}(t_k) + (f_u dt)\,\delta\mathbf{u}(t_k) \end{aligned} \tag{C.6}$$

Suppose that $(I_{n\times n} + f_x dt) = \Phi$ and $(f_u dt) = B$, then

$$\delta\mathbf{x}(t_{k+1}) = \Phi\delta\mathbf{x}(t_k) + B\delta\mathbf{u}(t_k) \tag{C.7}$$

C.1.2 The second order expansion of $Q(\mathbf{x}, \mathbf{u}, t)$ as a function of the running cost and the value function

Given that

$$Q(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) = L(\mathbf{x}(t_k), \mathbf{u}(t_k), t_k) + V(\mathbf{x}(t_k), t_k) \tag{C.8}$$

The left-hand side of Equation C.8 can be expanded as follows:

$$\begin{aligned} Q\left(\mathbf{x}\left(t_k\right),\mathbf{u}\left(t_k\right),t_k\right) &= Q\left(\mathbf{x}\left(t_k\right)-\bar{\mathbf{x}}\left(t_k\right)+\bar{\mathbf{x}}\left(t_k\right),\mathbf{u}\left(t_k\right)-\bar{\mathbf{u}}\left(t_k\right)+\bar{\mathbf{u}}\left(t_k\right),t_k\right) \\ &= Q\left(\bar{\mathbf{x}}\left(t_k\right)+\delta\mathbf{x}\left(t_k\right),\bar{\mathbf{u}}\left(t_k\right)+\delta\mathbf{u}\left(t_k\right),t_k\right) \\ &= Q\left(\bar{\mathbf{x}}\left(t_k\right),\bar{\mathbf{u}}\left(t_k\right),t_k\right)+\begin{bmatrix} Q_x & Q_u \end{bmatrix}^T \begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix} \\ &\quad +\tfrac{1}{2}\begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix} \end{aligned} \tag{C.9}$$

As such, the right-hand side of Equation C.8 can be expanded in the same fashion as follows:

$$\begin{aligned} L\left(\mathbf{x}\left(t_k\right),\mathbf{u}\left(t_k\right),t_k\right)+V\left(\mathbf{x}\left(t_{k+1}\right),t_{k+1}\right) &= L\left(\bar{\mathbf{x}}\left(t_k\right),\bar{\mathbf{u}}\left(t_k\right),t_k\right)+\begin{bmatrix} L_x & L_u \end{bmatrix}^T \begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix} \\ &\quad +\tfrac{1}{2}\begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix}^T \begin{bmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{bmatrix} \begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix} \\ &\quad +V\left(\bar{\mathbf{x}}\left(t_{k+1}\right),t_{k+1}\right)+V_x^T\delta\mathbf{x}\left(t_{k+1}\right) \\ &\quad +\tfrac{1}{2}\delta\mathbf{x}\left(t_{k+1}\right)^T V_{xx}\delta\mathbf{x}\left(t_{k+1}\right) \end{aligned} \tag{C.10}$$

Substitute the linearized dynamics in discrete time, Equation C.7, into Equation C.10:

$$\begin{aligned} L\left(\mathbf{x}\left(t_k\right),\mathbf{u}\left(t_k\right),t_k\right)+V\left(\mathbf{x}\left(t_{k+1}\right),t_{k+1}\right) &= L\left(\bar{\mathbf{x}}\left(t_k\right),\bar{\mathbf{u}}\left(t_k\right),t_k\right)+\begin{bmatrix} L_x & L_u \end{bmatrix}^T \begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix} \\ &\quad +\tfrac{1}{2}\begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix}^T \begin{bmatrix} L_{xx} & L_{xu} \\ L_{ux} & L_{uu} \end{bmatrix} \begin{Bmatrix} \delta\mathbf{x}\left(t_k\right) \\ \delta\mathbf{u}\left(t_k\right) \end{Bmatrix} \\ &\quad +V\left(\bar{\mathbf{x}}\left(t_{k+1}\right),t_{k+1}\right)+V_x^T\left(\Phi\delta\mathbf{x}\left(t_k\right)+B\delta\mathbf{u}\left(t_k\right)\right) \\ &\quad +\tfrac{1}{2}\left(\Phi\delta\mathbf{x}\left(t_k\right)+B\delta\mathbf{u}\left(t_k\right)\right)^T V_{xx}\left(\Phi\delta\mathbf{x}\left(t_k\right)+B\delta\mathbf{u}\left(t_k\right)\right) \end{aligned} \tag{C.11}$$

Now, equate Equation C.9 and C.11 in terms of $\delta\mathbf{x}\left(t_k\right)$ and $\delta\mathbf{u}\left(t_k\right)$, then

$$\begin{aligned} Q\left(\bar{\mathbf{x}}\left(t_k\right),\bar{\mathbf{u}}\left(t_k\right),t_k\right) &= L\left(\bar{\mathbf{x}}\left(t_k\right),\bar{\mathbf{u}}\left(t_k\right),t_k\right)+V\left(\bar{\mathbf{x}}\left(t_{k+1}\right),t_{k+1}\right) \\ Q_x &= L_x+\Phi^T V_x\left(t_{k+1}\right) \\ Q_u &= L_u+B^T V_x\left(t_{k+1}\right) \\ Q_{xx} &= L_{xx}+\Phi^T V_{xx}\left(t_{k+1}\right)\Phi \\ Q_{xu} &= L_{xu}+\Phi^T V_{xx}\left(t_{k+1}\right)B \\ Q_{ux} &= L_{ux}+B^T V_{xx}\left(t_{k+1}\right)\Phi \\ Q_{uu} &= L_{uu}+B^T V_{xx}\left(t_{k+1}\right)B \end{aligned} \tag{C.12}$$

C.1.3 Optimal control corrections for $\delta\mathbf{u}^*$

The optimal control corrections for $\delta\mathbf{u}^*$ can be found by taking the first partial derivative of Equation C.9 with respect to $\delta\mathbf{u}$ and making it to be zero.

$$\frac{\partial Q}{\partial\mathbf{u}}=Q_u^T+\frac{1}{2}\delta\mathbf{x}\left(t_k\right)^T Q_{xu}+\frac{1}{2}\delta\mathbf{x}\left(t_k\right)^T Q_{ux}^T+\delta\mathbf{u}^*\left(t_k\right)^T Q_{uu} \tag{C.13}$$

By knowing that $Q_{ux}^T = Q_{xu}$, Equation C.13 can be rewritten as follows:

$$\begin{aligned} \delta\mathbf{u}^*(t_k) &= -Q_{uu}^{-1}\left(Q_u + Q_{ux}\delta\mathbf{x}(t_k)\right) \\ &= \underbrace{-Q_{uu}^{-1}Q_u}_{\text{feedforward}}\underbrace{-Q_{uu}^{-1}Q_{ux}\delta\mathbf{x}(t_k)}_{\text{feedback}} \end{aligned} \tag{C.14}$$

C.1.4 Backward equations for $V(\bar{\mathbf{x}})$, V_x, and V_{xx}

By substituting Equation C.14 into Equation C.2, the $\min_{\mathbf{u}}$ operator can be removed and shown as follows:

$$\begin{aligned} V(\mathbf{x}(t_k), t_k) = \; & Q(\bar{\mathbf{x}}(t_k), \bar{\mathbf{u}}(t_k), t_k) \\ & + \begin{bmatrix} Q_x & Q_u \end{bmatrix}^T \begin{Bmatrix} \delta\mathbf{x}(t_k) \\ -Q_{uu}^{-1}\left(Q_u + Q_{ux}\delta\mathbf{x}(t_k)\right) \end{Bmatrix} \\ & + \tfrac{1}{2}\begin{Bmatrix} \delta\mathbf{x}(t_k) \\ -Q_{uu}^{-1}\left(Q_u + Q_{ux}\delta\mathbf{x}(t_k)\right) \end{Bmatrix}^T \begin{bmatrix} Q_{xx} & Q_{xu} \\ Q_{ux} & Q_{uu} \end{bmatrix} \begin{Bmatrix} \delta\mathbf{x}(t_k) \\ -Q_{uu}^{-1}\left(Q_u + Q_{ux}\delta\mathbf{x}(t_k)\right) \end{Bmatrix} \end{aligned} \tag{C.15}$$

Then, expand the left-hand side of Equation C.15 by the second order:

$$V(\mathbf{x}(t_k), t_k) = V(\bar{\mathbf{x}}(t_k), t_k) + V_x^T\delta\mathbf{x}(t_k) + \frac{1}{2}\delta\mathbf{x}(t_k)^T V_{xx}\delta\mathbf{x}(t_k) \tag{C.16}$$

Equate Equation C.15 and Equation C.16 in terms of the zeroth, first, and second order of $\delta\mathbf{x}(t_k)$:

$$\begin{aligned} V(\bar{\mathbf{x}}(t_k), t_k) &= Q(\bar{\mathbf{x}}(t_k), \bar{\mathbf{u}}(t_k), t_k) - \tfrac{1}{2}Q_u^T Q_{uu}^{-1}Q_u \\ V_x &= Q_x - Q_{xu}Q_{uu}^{-1}Q_u \\ V_{xx} &= Q_{xx} - Q_{xu}Q_{uu}^{-1}Q_{ux} \end{aligned} \tag{C.17}$$

C.2 Part II: DDP implementation

C.2.1 The inverted pendulum problem

Setup

The inverted pendulum dynamics:

$$I\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u \tag{C.18}$$

where $I = ml^2$ is the inertia, m is the mass, $g = 9.81\frac{m}{\text{sec}^2}$ is the gravitational acceleration, l is the length, b is damping and u is the control torque. The initial states are $\theta = \dot{\theta} = 0$, and the final desired states are $\theta = \pi$ & $\dot{\theta} = 0$.

Suppose that:

$$\mathbf{x} = \begin{Bmatrix} \theta \\ \dot{\theta} \end{Bmatrix}, \quad \mathbf{x}_{\text{final}} = \begin{Bmatrix} \pi \\ 0 \end{Bmatrix} \tag{C.19}$$

Then,

$$\dot{\mathbf{x}} = \begin{Bmatrix} \dot{\theta} \\ \ddot{\theta} \end{Bmatrix} = \begin{Bmatrix} \dot{\theta} \\ -\frac{b}{I}\dot{\theta} - \frac{mgl}{I}\sin\theta \end{Bmatrix} + \begin{Bmatrix} 0 \\ \frac{1}{I} \end{Bmatrix} u \tag{C.20}$$

Recall Equation C.6 and C.7,

$$\begin{aligned} f_x &= \begin{bmatrix} 0 & 1 \\ -\frac{mgl}{I} & -\frac{b}{I} \end{bmatrix} \\ f_u &= \begin{Bmatrix} 0 \\ \frac{1}{I} \end{Bmatrix} \end{aligned} \tag{C.21}$$

$$\begin{aligned} \Phi &= I_{2\times 2} + f_x dt \\ B &= f_u dt \end{aligned} \tag{C.22}$$

Let's define the terminal and running cost functions. In order to make them convex, quadratic forms in terms of **x** and **u** are appropriate.

$$\begin{aligned} \phi\left(\mathbf{x}\left(t_f\right), t_f\right) &= \tfrac{1}{2}\left(\mathbf{x}\left(t_f\right) - \mathbf{x}_{\text{final}}\right)^T K_f \left(\mathbf{x}\left(t_f\right) - \mathbf{x}_{\text{final}}\right) \\ L\left(\mathbf{x}\left(t_k\right), \mathbf{u}\left(t_k\right), t_k\right) &= \left(\tfrac{1}{2}\mathbf{x}\left(t_k\right)^T K \mathbf{x}\left(t_k\right) + \tfrac{1}{2}\mathbf{u}\left(t_k\right)^T R \mathbf{u}\left(t_k\right)\right) dt \end{aligned} \tag{C.23}$$

where K_f, K, and R are control gains.

Results

The inverted pendulum problem was implemented and simulated in Matlab. The designed gains are:

$$\begin{aligned} K_f &= \begin{bmatrix} 30 & 0 \\ 0 & 30 \end{bmatrix} \\ K &= \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \\ R &= 10 \end{aligned} \tag{C.24}$$

The simulated time horizon was three seconds, and the time horizon was discretized into 20 segments. Figure C.1 and C.2 depict the results of DDP process. It shows that the pendulum pushed slightly forward first, and then backward, and finally forward again.

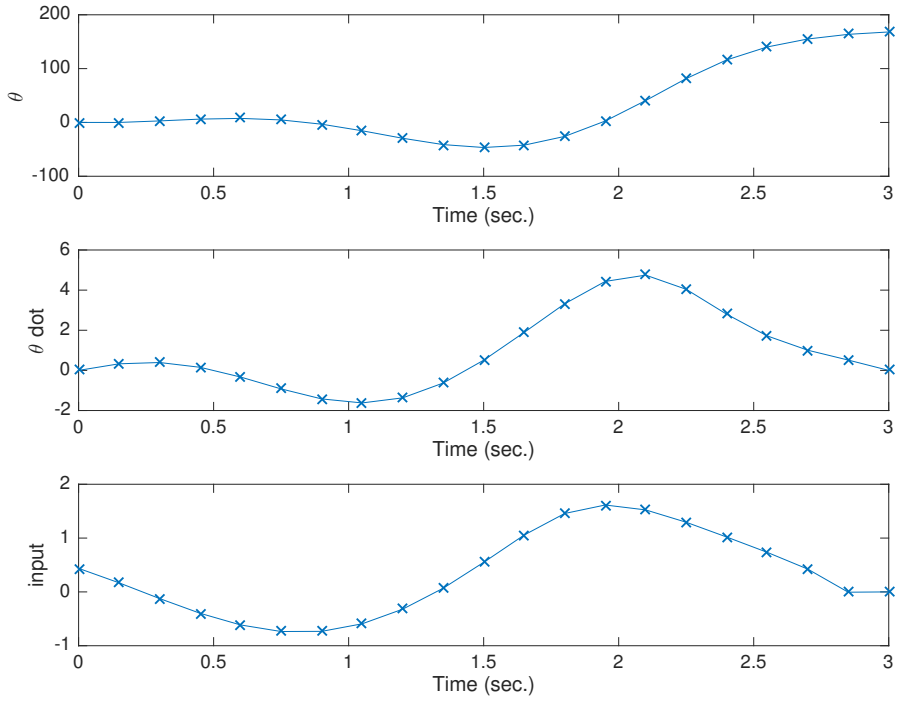

Figure C.1: Inverted pendulum simulation results - states

C.2.2 The cart pole problem

Setup

The cart pole dynamics:

$$\begin{aligned} \ddot{x} &= \frac{1}{m_c + m_p \sin^2\theta}\left(f + m_p \sin\theta\left(l\dot{\theta}^2 + g\cos\theta\right)\right) \\ \ddot{\theta} &= \frac{1}{l\left(m_c + m_p \sin^2\theta\right)}\left(-f\cos\theta - m_p l\dot{\theta}^2\cos\theta\sin\theta - (m_c + m_p)\, g\sin\theta\right) \end{aligned} \tag{C.25}$$

where f is the control, $g = 9.81\frac{m}{\text{sec}^2}$ is the gravitational acceleration, $m_c = 1.0kg$ is the mass of the cart, $m_p = 0.01kg$ is the mass of the pole, and $l = 0.25$ is the length of the pole. The initial states are $x = \dot{x} = \theta = \dot{\theta} = 0$, and the final desired states are $\dot{x} = \dot{\theta} = 0$ and $\theta = \pi$.

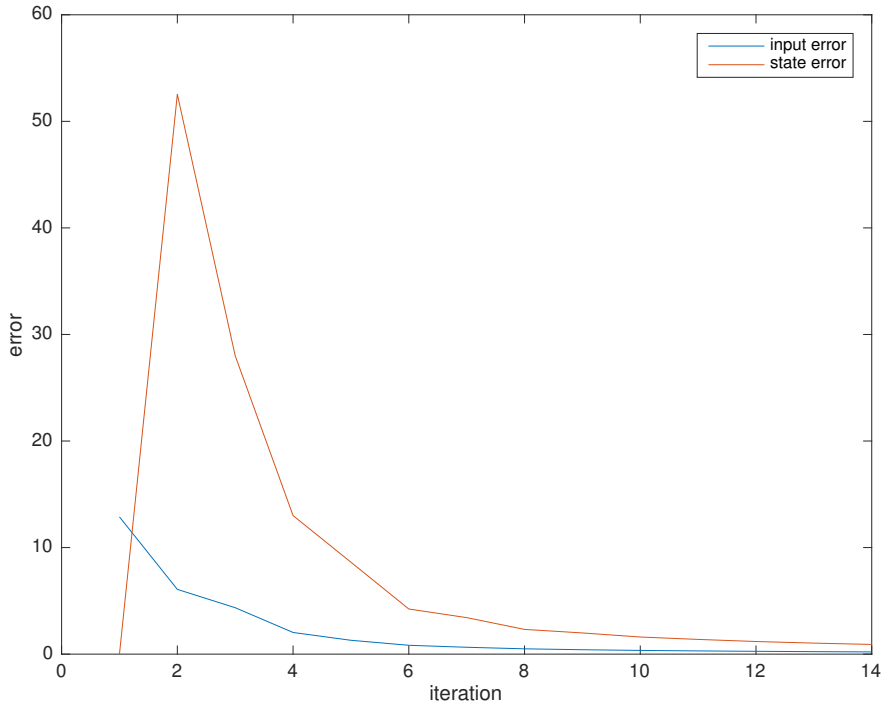

Figure C.2: Inverted pendulum simulation results - convergence

Suppose that:

$$\mathbf{x} = \begin{Bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{Bmatrix}, \quad \mathbf{x}_{\text{final}} = \begin{Bmatrix} \text{free} \\ 0 \\ \pi \\ 0 \end{Bmatrix}, \text{and} \quad u = f \tag{C.26}$$

Then,

$$
\begin{aligned}
\dot{\mathbf{x}} &= \begin{Bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{Bmatrix} \\
&= \begin{Bmatrix} \dot{x} \\ \frac{1}{m_c + m_p \sin^2\theta} \left(m_p \sin\theta \left(l\dot{\theta}^2 + g\cos\theta \right) \right) \\ \dot{\theta} \\ \frac{1}{l\left(m_c + m_p \sin^2\theta\right)} \left(-m_p l \dot{\theta}^2 \cos\theta \sin\theta - \left(m_c + m_p\right) g \sin\theta \right) \end{Bmatrix} \\
&\quad + \begin{Bmatrix} 0 \\ \frac{1}{m_c + m_p \sin^2\theta} \\ 0 \\ -\frac{1}{l\left(m_c + m_p \sin^2\theta\right)} \cos\theta \end{Bmatrix} u
\end{aligned}
\tag{C.27}
$$

Recall Equation C.6 and C.7,

$$
\begin{aligned}
f_x &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & f_{23} & \frac{1}{m_c + m_p \sin^2\theta} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & f_{43} & \frac{-\cos\theta}{l\left(m_c + m_p \sin^2\theta\right)} \end{bmatrix} \\
f_u &= \begin{Bmatrix} 0 \\ \frac{1}{m_c + m_p \sin^2\theta} \\ 0 \\ -\frac{1}{l\left(m_c + m_p \sin^2\theta\right)} \cos\theta \end{Bmatrix}
\end{aligned}
\tag{C.28}
$$

where

$$
\begin{aligned}
f_{23} &= -\frac{2m_p u \sin\theta\cos\theta}{\left(m_c+m_p\sin^2\theta\right)^2} + \frac{m_p l\dot{\theta}^2\cos\theta\left(m_c-m_p\sin^2\theta\right)}{\left(m_c+m_p\sin^2\theta\right)^2} + \frac{2m_p g((2m_c+m_p)\cos 2\theta-m_p)}{(2m_c-m_p\cos 2\theta+m_p)^2} \\
f_{43} &= \frac{u\sin\theta-g(m_c+m_p)\cos\theta}{l\left(m_c+m_p\sin^2\theta\right)} + \frac{2m_p f\sin\theta\cos^2\theta+2m_p g(m_c+m_p)\sin^2\theta\cos\theta}{l\left(m_c+m_p\sin^2\theta\right)^2} + \frac{2m_p^2\dot{\theta}^2\sin^2\theta\cos^2\theta}{\left(m_c+m_p\sin^2\theta\right)^2} + \frac{m_p\dot{\theta}^2\left(\sin^2\theta-\cos^2\theta\right)}{m_c+m_p\sin^2\theta}
\end{aligned}
\tag{C.29}
$$

Then,

$$
\begin{aligned}
\Phi &= I_{4\times 4} + f_x dt \\
B &= f_u dt
\end{aligned}
\tag{C.30}
$$

Finally, define the terminal and running cost functions. In order to make them convex, quadratic forms in terms of **x** and **u** are appropriate.

$$
\begin{aligned}
\phi\left(\mathbf{x}\left(t_f\right), t_f\right) &= \tfrac{1}{2}\left(\mathbf{x}\left(t_f\right)-\mathbf{x}_{\text{final}}\right)^T K_f\left(\mathbf{x}\left(t_f\right)-\mathbf{x}_{\text{final}}\right) \\
L\left(\mathbf{x}\left(t_k\right), \mathbf{u}\left(t_k\right), t_k\right) &= \left(\tfrac{1}{2}\mathbf{x}\left(t_k\right)^T K\mathbf{x}\left(t_k\right) + \tfrac{1}{2}\mathbf{u}\left(t_k\right)^T R\mathbf{u}\left(t_k\right)\right) dt
\end{aligned}
\tag{C.31}
$$

where K_f, K, and R are control gains.

Results

The cart-pole problem was implemented and simulated in Matlab. The designed gains are:

$$\begin{aligned} K_f &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 300 & 0 & 0 \\ 0 & 0 & 300 & 0 \\ 0 & 0 & 0 & 300 \end{bmatrix} \\ K &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ R &= 10 \end{aligned} \tag{C.32}$$

The simulated time horizon was two seconds, and the time horizon was discretized into 20 segments. Figure C.3 depicts the state and control results of DDP process, and Figure C.4 shows its convergence behavior.

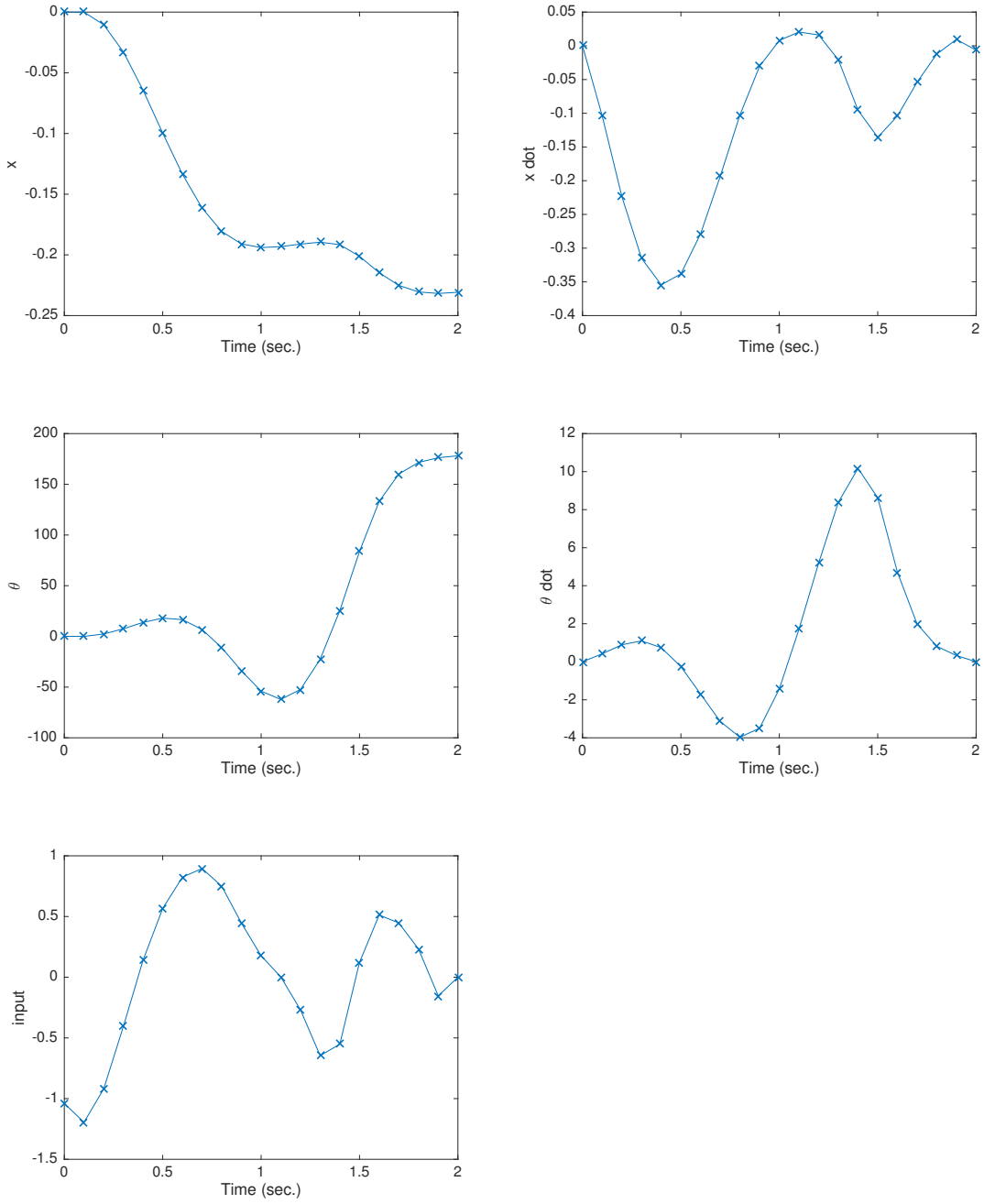

Figure C.3: Cart-pole problem results

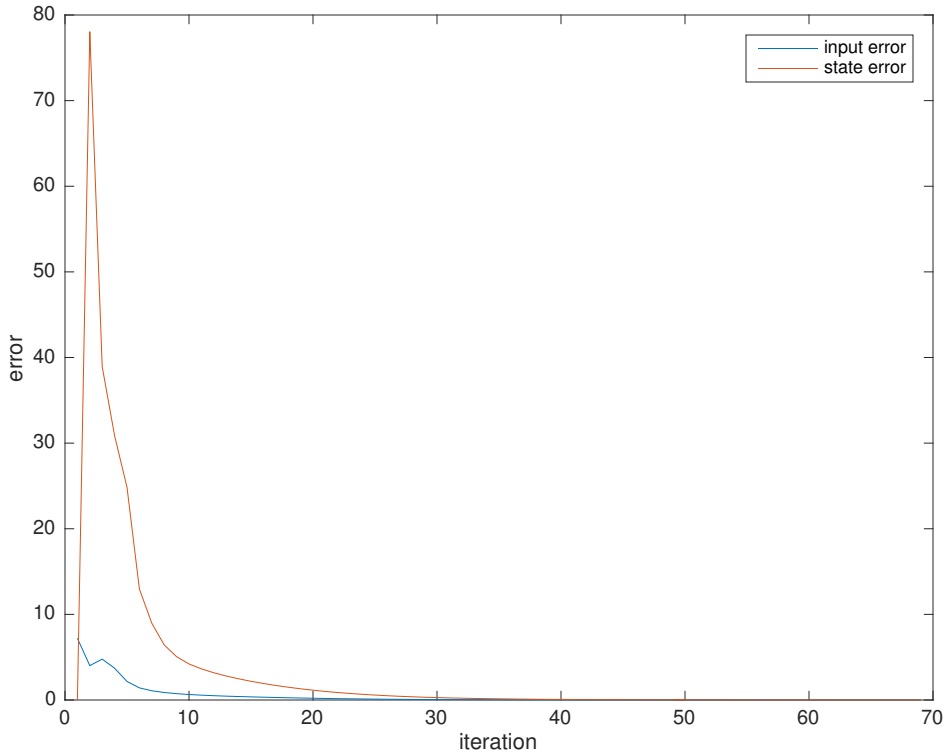

Figure C.4: Cart-pole problem convergence behavior

REFERENCES

[1] "Global military uav market 2018-2028," GlobalData, Article, 2018.

[2] "Uav drones - global market outlook (2016-2022)," Statistics MRC, Article, 2016.

[3] J. Camhi, "The drones report," Business Insider, Article, 2016.

[4] A. Bhutani and P. Wadhwani, "Military drone market worth over $13bn by 2024," Global Market Insights, article, 2018.

[5] "Commercial drones in 2022 - our predictions," Interact Analysis, article.

[6] "Operation and certification of small unmanned aircraft systems," Federal Aviation Administration, Regulation, 2016.

[7] L. Downes, "Whats wrong with the faas new drone rules," Harvard Business Review, Article, 2015.

[8] E. Chow, A. Cuadra, and C. Whitlock, "Fallen from the skies," The Washington Post, Article, 2016.

[9] C. Whitlock, "How crashing drones are exposing secrets about u.s. war operations," The Washington Post, Article, 2015.

[10] ——, "More air force drones are crashing than ever as mysterious new problems emerge," The Washington Post, Article, 2016.

[11] K. W. Williams, "A summary of unmanned aircraft accident/incident data: Human factors implications," DOT/FAA/AM, Tech. Rep., 2016.

[12] C. Whitlock, "Crashes mount as military flies more drones in u.s.," The Washington Post, Article, 2014.

[13] M. G. Balchanos, "A probabilistic technique for the assessment of complex dynamic system resilience," PhD thesis, Georgia Institute of Technology, 2012.

[14] M. Couture, "Complexity and chaos-state-of-the-art; formulations and measures of complexity," DTIC Document, Tech. Rep., 2007.

[15] H. H. Shelton, *Joint vision 2020*. US Government Printing Office, 2000.

[16] P. Anderson, "Perspective: Complexity theory and organization science," *Organization science*, vol. 10, no. 3, pp. 216–232, 1999.

[17] C. S. Holling, "Resilience and stability of ecological systems," *Annual review of ecology and systematics*, pp. 1–23, 1973.

[18] S Simant, C Fantuzzi, and R Patton, *Model-based fault diagnosis in dynamic systems using identifeicatio techniques*, 2002.

[19] M. Witczak, *Modeling and estimation strategies for fault diagnosis of non-linear systems, part 1*.

[20] P. Patrón, E. Miguelanez, Y. R. Petillot, D. M. Lane, and J. Salvi, "Adaptive mission plan diagnosis and repair for fault recovery in autonomous underwater vehicles," in *Oceans 2008*, IEEE, 2008, pp. 1–9.

[21] E. Sobhani-Tehrani and K. Khorasani, *Fault diagnosis of nonlinear systems using a hybrid approach*. Springer Science & Business Media, 2009, vol. 383.

[22] "Military standard: Procedures for performing a failure mode, effects, and criticality analysis," Department of Defense, Washington, D.C., Standard, 1980.

[23] "British standard: Reliability of systems, equipment and components - part 5: Guide to failure modes, effects and criticality analysis (fmea and fmeca)," British Standards Institute, Standard, 1991.

[24] E. Sobhani-Tehrani and K. Khorasani, *Fault diagnosis of nonlinear systems using a hybrid approach*. New York: Springer, 2009, vol. 383.

[25] G. Vachtsevanos, F. L. Lewis, M. Roemer, A. Hess, and B. Wu, *Intelligent fault diagnosis and prognosis for engineering systems*. Hoboken, New Jersey: John Wiley and Sons, Inc., 2006.

[26] V. Venkatasubramanian, R. Rengaswamy, K. Yin, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part i: Quantitative model-based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 293–311, 2003.

[27] R. J. Patton, *Model-based fault diagnosis, part 1*, Lecture note.

[28] F. Lackinger and W. Nejdl, "Integrating model-based monitoring and diagnosis of complex dynamic systems.," in *Ijcai*, Citeseer, 1991, pp. 1123–1128.

[29] V. Venkatasubramanian, R. Rengaswamy, and S. N. Kavuri, "A review of process fault detection and diagnosis: Part ii: Qualitative models and search strategies," *Computers & chemical engineering*, vol. 27, no. 3, pp. 313–326, 2003.

[30] V. Venkatasubramanian, R. Rengaswamy, S. N. Kavuri, and K. Yin, "A review of process fault detection and diagnosis: Part iii: Process history based methods," *Computers & chemical engineering*, vol. 27, no. 3, pp. 327–346, 2003.

[31] R. J. Patton, *Model-based fault diagnosis, part 1*, Lecture note.

[32] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel, "Prognostics and health management design for rotary machinery systemsreviews, methodology and applications," *Mechanical systems and signal processing*, vol. 42, no. 1-2, pp. 314–334, 2014.

[33] M. E. Orchard, "A particle filtering-based framework for on-line fault diagnosis and failure prognosis," PhD thesis, Georgia Institute of Technology, 2007.

[34] J.-P. Watson, R. Guttromson, C. Silva-Monroy, R. Jeffers, K. Jones, J. Ellison, C. Rath, J. Gearhart, D. Jones, T. Corbet, *et al.*, "Conceptual framework for developing resilience metrics for the electricity oil and gas sectors in the united states," *Sandia national laboratories, albuquerque, nm (united states), tech. rep*, 2014.

[35] *Caib's final report*, 2003.

[36] K. W. Williams, "A summary of unmanned aircraft accident/incident data: Human factors implications," DTIC Document, Tech. Rep., 2004.

[37] E. Hollnagel, "Resilience engineering: Why, what, and how," in *The 17th nordic research conference on safety*, 2007.

[38] A. M. Madni and S. Jackson, "Towards a conceptual framework for resilience engineering," *Systems journal, ieee*, vol. 3, no. 2, pp. 181–191, 2009.

[39] P. P. Directive, *Critical infrastructure security and resilience. ppd-21, released february 12, 2013*, 2013.

[40] E. Hollnagel, D. D. Woods, and N. Leveson, *Resilience engineering: Concepts and precepts*. Ashgate Publishing, Ltd., 2007.

[41] W. F. Stevens, "Weapon system effectiveness industry advisory committee," SAE Technical Paper, Tech. Rep., 1964.

[42] M. G. Richards, A. M. Ross, D. E. Hastings, and D. H. Rhodes, "Multi-attribute tradespace exploration for survivability," PhD thesis, Massachusetts Institute of Technology, Engineering Systems Division, 2009.

[43] E. D. Vugrin, D. E. Warren, M. A. Ehlen, and R. C. Camphouse, "A framework for assessing the resilience of infrastructure and economic systems," in *Sustainable and resilient critical infrastructure systems*, Springer, 2010, pp. 77–116.

[44] J Rasmussen and I Svedung, "Proactive risk management in a dynamic society karlstad," *Sweden: Swedish rescue services agency*, 2000.

[45] N. Leveson, "A new accident model for engineering safer systems," *Safety science*, vol. 42, no. 4, pp. 237–270, 2004.

[46] M. G. Balchanos, "A probabilistic technique for the assessment of complex dynamic system resilience," 2012.

[47] R. E. Weibel and R. J. Hansman, "Safety considerations for operation of unmanned aerial vehicles in the national airspace system," 2006.

[48] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *Control systems technology, ieee transactions on*, vol. 18, no. 3, pp. 636–653, 2010.

[49] Y. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control systems," *Annual reviews in control*, vol. 32, no. 2, pp. 229–252, 2008.

[50] M. Steinberg, "Historical overview of research in reconfigurable flight control," *Proceedings of the institution of mechanical engineers, part g: Journal of aerospace engineering*, vol. 219, no. 4, pp. 263–275, 2005.

[51] J. McMahan, "Flight 1080," *Air line pilot*, vol. 47, no. 7, pp. 6–11, 1978.

[52] R. J. Montoya, W. Howell, W. Bundick, A. Ostroff, R. Hueschen, and C. M. Belcastro, "Restructurable controls," 1983.

[53] J. Maciejowski and C. Jones, "Mpc fault-tolerant flight control case study: Flight 1862," in *Proceedings of the international federation of automatic control on safeprocess sympoisum*, 2003, pp. 119–124.

[54] J. S. Eterno, J. L. Weiss, D. P. Looze, and A. Willsky, "Design issues for fault tolerant-restructurable aircraft control," in *Decision and control, 1985 24th ieee conference on*, IEEE, 1985, pp. 900–905.

[55] P. S. Maybeck and R. D. Stevens, "Reconfigurable flight control via multiple model adaptive control methods," *Aerospace and electronic systems, ieee transactions on*, vol. 27, no. 3, pp. 470–480, 1991.

[56] D. D. Moerder, N. Halyo, J. R. Broussard, and A. K. Caglayan, "Application of precomputed control laws in a reconfigurable aircraftflight control system," *Journal of guidance, control, and dynamics*, vol. 12, no. 3, pp. 325–333, 1989.

[57] Y. Zhang and J. Jiang, "Integrated active fault-tolerant control using imm approach," *Aerospace and electronic systems, ieee transactions on*, vol. 37, no. 4, pp. 1221–1235, 2001.

[58] R. J. Patton, "Fault-tolerant control systems: The 1997 situation," in *Ifac symposium on fault detection supervision and safety for technical processes*, vol. 3, 1997, pp. 1033–1054.

[59] Y. Zhang and J. Jiang, "Active fault-tolerant control system against partial actuator failures," *Iee proceedings-control theory and applications*, vol. 149, no. 1, pp. 95–104, 2002.

[60] G. R. Drozeski, "A fault-tolerant control architecture for unmanned aerial vehicles," PhD thesis, 2005.

[61] N. S. Clements, "Fault tolerant control of complex dynamical systems," PhD thesis, Georgia Institute of Technology, 2003.

[62] J. Ge, G. Kacprzynski, M. Roemer, and G. Vachtsevanos, "Automated contingency management design for uavs," in *Aiaa 1st intelligent systems technical conference*, 2004, p. 6464.

[63] L. Tang, G. J. Kacprzynski, K. Goebel, A. Saxena, B. Saha, and G. Vachtsevanos, "Prognostics-enhanced automated contingency management for advanced autonomous systems," in *2008 international conference on prognostics and health management*, IEEE, 2008, pp. 1–9.

[64] L. Tang, E. Hettler, B. Zhang, and J. DeCastro, "A testbed for real-time autonomous vehicle phm and contingency management applications," in *Annual conference of the prognostics and health management society*, 2011, pp. 1–11.

[65] D. W. Brown, G. Georgoulas, B. Bole, H.-L. Pei, M Orchard, L Tang, B Saha, A Saxena, K Goebel, and G Vachtsevanos, "Prognostics enhanced reconfigurable control of electro-mechanical actuators," in *Annual conference of the prognostics and health management society*, 2009.

[66] B. M. Bole, "Load allocation for optimal risk management in systems with incipient failure modes," PhD thesis, Georgia Institute of Technology, 2013.

[67] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 ieee international conference on robotics and automation (icra)*, IEEE, 2014, pp. 1168–1175.

[68] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *Ai communications*, vol. 7, no. 1, pp. 39–59, 1994.

[69] M. M. Richter, "Knowledge containers," *Readings in case-based reasoning. morgan kaufmann publishers*, 2003.

[70] ——, "The search for knowledge, contexts, and case-based reasoning," *Engineering applications of artificial intelligence*, vol. 22, no. 1, pp. 3–9, 2009.

[71] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd acm national conference*, ACM, 1968, pp. 517–524.

[72] J. Lunze and J. H. Richter, "Reconfigurable fault-tolerant control: A tutorial introduction," *European journal of control*, vol. 14, no. 5, pp. 359–386, 2008.

[73] M Staroswiecki, "Fault tolerant control: The pseudo-inverse method revisited," *Ifac proceedings volumes*, vol. 38, no. 1, pp. 418–423, 2005.

[74] S. M. de Oca, V. Puig, D. Theillio, and S. Tornil-Sin, "Fault-tolerant control design using lpv admissible model matching: Application to a two-degree of freedom helicopter," in *2009 17th mediterranean conference on control and automation*, IEEE, 2009, pp. 522–527.

[75] Z. Gao and P. J. ANTSAKLIS, "Reconfigurable control system design via perfect model following," *International journal of control*, vol. 56, no. 4, pp. 783–798, 1992.

[76] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[77] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Conditions under which suboptimal nonlinear mpc is inherently robust," *Systems & control letters*, vol. 60, no. 9, pp. 747–755, 2011.

[78] G. Pannocchia, *Course on model predictive control part iii stability and robustness*.

[79] C. N. Jones and J. Maciejowski, "Reconfigurable flight control first year report," *Department of engineering, university of cambridge*, 2005.

[80] R. Bellman, "Dynamic programming, princeton, nj: Princeton univ," *Versity press. bellmandynamic programming1957*, 1957.

[81] M. Kobilarov, D.-N. Ta, and F. Dellaert, "Differential dynamic programming for optimal estimation," in *2015 ieee international conference on robotics and automation (icra)*, IEEE, 2015, pp. 863–869.

[82] R. S. Sutton, A. G. Barto, *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 135.

[83] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

[84] D. Silver, *Reinforcment learning*, Lecture note.

[85] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[86] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Icml*, 2014.

[87] A. Irpan, *Deep reinforcement learning doesn't work yet*, `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.

[88] R. Akerkar and P. Sajja, *Knowledge-based systems*. Sudbury, MA, USA: Jones & Bartlett.

[89] A. Saxena, "Knowledge-based architecture for integrated condition based maintenance of engineering systems," PhD thesis, Georgia Institute of Technology, 2007.

[90] E. Shortliffe, "Mycin: Computer-based consultations in medical therapeutics," *American elservier, new york*, 1976.

[91] S. Slade, "Case-based reasoning: A research paradigm," *Ai magazine*, vol. 12, no. 1, pp. 42–42, 1991.

[92] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat, "Building expert system," 1983.

[93] K. L. Myers, "Procedural reasoning system users guide," *Artificial intelligence center, sri international, menlo park, ca, usa*, vol. 1, 1997.

[94] M. P. Georgeff and F. F. Ingrand, "Monitoring and control of spacecraft systems using procedural reasoning," 1990.

[95] F. Ingrand and O. Despouys, "Extending procedural reasoning toward robot actions planning," in *Proceedings 2001 icra. ieee international conference on robotics and automation (cat. no. 01ch37164)*, IEEE, vol. 1, 2001, pp. 9–14.

[96] K. Kim, Y. Lee, S. Oh, D. Moroniti, D. Mavris, G. J. Vachtsevanos, N. Papamarkos, and G. Georgoulas, "Guidance, navigation, and control of an unmanned hovercraft," in *Control & automation (med), 2013 21st mediterranean conference on*, IEEE, 2013, pp. 380–387.

[97] C. Sconyers, Y. Lee, K. Kim, S. Oh, D. Mavris, N. Oza, R. Mah, R. Martin, I. A. Raptis, and G. J. Vachtsevanos, "Diagnosis of fault modes masked by control loops with an application to autonomous hovercraft systems," *International journal of prognostics and health management*, 2013.

[98] L. Zaccarian, "Dc motors: Dynamic model and control techniques," *Lecture notes., roma, italy*, 2012.

[99] A. De Martin, G. Jacazio, and G. Vachtsevanos, "Anomaly detection and prognosis for primary flight control emas," in *3rd european conference of the prognostics and healh management society*, 2016, pp. 5–8.

[100] S. Nandi, H. A. Toliyat, and X. Li, "Condition monitoring and fault diagnosis of electrical motorsa review," *Ieee transactions on energy conversion*, vol. 20, no. 4, pp. 719–729, 2005.

[101] *Winding shorted turn-to-turn*, Online resource, The Electrical Apparatus Service Association, Inc. (EASA).

[102] B. Bole, L. Tang, K. Goebel, and G. Vachtsevanos, "Adaptive load-allocation for prognosis-based risk management," in *Annual conference of the prognostics and health management society*, 2011, pp. 1–10.

[103] R. P. Brent, "An algorithm with guaranteed convergence for finding a zero of a function," *The computer journal*, vol. 14, no. 4, pp. 422–425, 1971.

[104] I. F. Hooks and K. A. Farry, *Customer-centered products: Creating successful products through smart requirements management*. Amacom Books, 2001.

[105] R. T. Anderson, "Reliability design handbook," RELIABILITY ANALYSIS CENTER GRIFFISS AFB NY, Tech. Rep., 1976.

[106] T. Bedford, R. Cooke, *et al.*, *Probabilistic risk analysis: Foundations and methods*. Cambridge University Press, 2001.

[107] J. Dunjó, V. Fthenakis, J. A. Vílchez, and J. Arnaldos, "Hazard and operability (hazop) analysis. a literature review," *Journal of hazardous materials*, vol. 173, no. 1-3, pp. 19–32, 2010.

[108] T. Söderström and P. Stoica, "System identification," 1989.

[109] E. J. Vladislavleva, G. F. Smits, and D. Den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming," *Ieee transactions on evolutionary computation*, vol. 13, no. 2, pp. 333–349, 2008.

[110] *Eureqa: The a.i.-powered modeling engine*, Online resource, Nutonian, inc.

[111] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: An open-source robot operating system," in *Icra workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.

[112] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 ieee/rsj international conference on intelligent robots and systems (iros)(ieee cat. no. 04ch37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.

[113] R. Smith *et al.*, "Open dynamics engine," 2005.

[114] *Standard practice: System safety*, 2012.

[115] D. S. Soban, "A methodology for the probabilistic assessment of system effectiveness as applied to aircraft survivability and susceptibility," PhD thesis, Citeseer, 2001.

[116] D. A. Rains, "Combatant ship design guidance through mission effectiveness analysis," *Naval engineers journal*, vol. 96, no. 3, pp. 112–127, 1984.

[117] W. C. Christensen and F. A. Manuele, *Safety through design*. Amer Society of Mechanical, 1999.

[118] B. Moriarty, *System safety engineering and management*. John Wiley & Sons, 1990.

[119] D. S. Soban and D. N. Mavris, "The need for a military system effectiveness framework: The system of systems approach," in *1st aiaa, aircraft, technology integration, and operations forum, los angeles, ca*, 2001.

[120] N. G. Leveson, *Safeware: System safety and computers*. New York, NY, USA: ACM, 1995, ISBN: 0-201-11972-2.

[121] S. Sklet, "Safety barriers: Definition, classification, and performance," *Journal of loss prevention in the process industries*, vol. 19, no. 5, pp. 494–506, 2006.

[122] R. E. Ball and D. B. Atkinson, "A history of the survivability design of military aircraft," DTIC Document, Tech. Rep., 1998.

[123] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, T. A. Longstaff, and N. R. Mead, "An approach to survivable systems," in *Nato ist symposium on protecting information systems in the 21st century*, 1999, pp. 754–759.

[124] J. C. Knight and K. J. Sullivan, "On the definition of survivability," *University of virginia, department of computer science, technical report cs-tr-33-00*, 2000.

[125] R. E. Ball, *The fundamentals of aircraft combat survivability analysis and design*. AIAA (American Institute of Aeronautics & Astronautics), 2003.

[126] S. Kaplan, "The words of risk analysis," *Risk analysis*, vol. 17, no. 4, pp. 407–417, 1997.

[127] W. Keller and M. Modarres, "A historical overview of probabilistic risk assessment development and its use in the nuclear power industry: A tribute to the late professor norman carl rasmussen," *Reliability engineering & system safety*, vol. 89, no. 3, pp. 271–285, 2005.

[128] B. D. Youn, K. Choi, R.-J. Yang, and L Gu, "Reliability-based design optimization for crashworthiness of vehicle side impact," *Structural and multidisciplinary optimization*, vol. 26, no. 3-4, pp. 272–283, 2004.

[129] G. E. Apostolakis, "How useful is quantitative risk assessment?" *Risk analysis*, vol. 24, no. 3, pp. 515–520, 2004.

[130] S. Kaplan and B. J. Garrick, "On the quantitative definition of risk," *Risk analysis*, vol. 1, no. 1, pp. 11–27, 1981.

[131] R. Foote, "Mathematics and complex systems," *Science*, vol. 318, no. 5849, pp. 410–412, 2007.

[132] J. M. Carlson and J. Doyle, "Complexity and robustness," *Proceedings of the national academy of sciences*, vol. 99, no. suppl 1, pp. 2538–2545, 2002.

[133] M. Rausand and K. Øien, "The basic concepts of failure analysis," *Reliability engineering & system safety*, vol. 53, no. 1, pp. 73–83, 1996.

[134] C. A. Ericson and C. Ll, "Fault tree analysis," in *System safety conference, orlando, florida*, 1999, pp. 1–9.

[135] M. Stamatelatos, H. Dezfuli, G. Apostolakis, C. Everline, S. Guarro, D. Mathias, A. Mosleh, T. Paulos, D. Riha, C. Smith, *et al.*, "Probabilistic risk assessment procedures guide for NASA managers and practitioners," 2011.

[136] A. Brown and D. Chen, "Probabilistic method for predicting ship collision damage," *Ocean engineering international journal*, vol. 6, no. 1, pp. 54–65, 2002.

[137] W. Bogard, *The bhopal tragedy: Language, logic, and politics in the production of a hazard*. Westview Press, 1989.

[138] S. Sklet, "Comparison of some selected methods for accident investigation," *Journal of hazardous materials*, vol. 111, no. 1, pp. 29–37, 2004.

[139] M. Rausand, "Reliability centered maintenance," *Reliability engineering & system safety*, vol. 60, no. 2, pp. 121–132, 1998.

[140] G. Taguchi, *Introduction to quality engineering: Designing quality into products and processes*. 1986.

[141] R. J. Ellison, D. A. Fisher, R. C. Linger, H. F. Lipson, and T. Longstaff, "Survivable network systems: An emerging discipline," DTIC Document, Tech. Rep., 1997.

[142] A. M. Ross, "Managing unarticulated value: Changeability in multi-attribute tradespace exploration," PhD thesis, Massachusetts Institute of Technology, Engineering Systems Division, 2006.

[143] C. Schwarz and H Drake, "Survivability Engineering, vol. 4, Aerospace Systems Survivability handbook Series, Joint Technical Coordinating Group on Aircraft Survivability (JTCG/AS)," JTCG/AS-01-D-005, Arlington, VA, Tech. Rep., 2001.

[144] U. Navy, "Us navy survavibility design handbook for surface ships. chief of naval operations ship safety and survivability office," OPNAV P-86-4-99 ed, Tech. Rep., 2000.

VITA

Sehwan Oh was born in Seoul, South Korea in 1981. He majored in mechanical engineering at Hanyang University and graduated in 2006.

He started his career from Korea Aerospace Industries in 2006, the same year of his graduation from college. He worked for the T-50 design team, especially for the manufacturing phase design of wing structures. The work experiences made him curious about where all the technical details came from and what made the aircraft look like the way it looks nowadays. His curiosity had him decide to change his life by taking a chance to study in the United States of America.

He joined the Aerospace Systems Design Laboratory at the Georgia Institute of Technology in 2008. Under the supervision of Dr. Mavris, he joined various research as a graduate research assistant. In the Rolls-Royce project, a practical data exploration and regression problems were studied. The following year, by switching the research topic, He joined the Transformable Craft research project sponsored by the Office of Naval Research (ONR) group. This research came to the development of a scaled autonomously operable hovercraft hardware and its simulation model. Eventually, this experience became a small seed of his thesis topic, and at the end of the long journey, he completed his Ph.D requirements in 2019. In the meantime, he researched on data processing and simulation model development for the smart campus and living building project sponsored by Georgia Institute of Technology.